MM CA1
DTC800
23
T20
c.2

TP 9254E

DEVELOPMENT OF A DERAILMENT ACCIDENT COMPUTER SIMULATION MODEL

Prepared for the Transportation Development Centre
and
Transport of Dangerous Goods Transport Canada

Date Due

NOV 8 [illegible]

BRODART, INC Cat No 23 233 Printed in U S

Transport Canada / Transports Canada

by

A.J. Coppens

and

J.D.E. Wong

W.R. Davis Engineering Limited

and

A.M. Birk

A.M. Birk Engineering

June 1988

416273

DDN 395536

89-05279/

The contents of this report reflect the views of W.R. Davis Engineering Limited and not necessarily the official views, or opinions of Transport Canada.

Un sommaire en français de ce document
est inclus avant la table des matières.

Transport Canada Transports Canada

Development Développement

REPORT DOCUMENTATION FORM

1. Transport Canada Report No.
TP 9254E

2. TDC Project No.
6835/36/37

3. Recipient's Catalogue No.

4. Title and Subtitle
Development of a Derailment Accident Computer Simulation Model

5. Report Date
June 1988

6. Performing Organization Report No.

7. Author(s)
A.J. Coppens, J.D.E. Wong, A. Bibby, A.M. Birk, R.J. Anderson

8. Transport Canada File No.
D1465-503

9. Performing Organization Name and Address
W.R. Davis Engineering Ltd.
1260 Old Innes Road
Ottawa, Ontario
K1B 3V3

10. DSS File No.
48SZ.T8200-6-6550

11. DSS or Transport Canada Contract No.
T8200-6-6550/01-SZ

12. Sponsoring Agency Name and Address

Transportation Development Centre(TDC)
200 René-Lévesque Blvd. West
Suite 601, West Tower
Montreal, Quebec
H2Z 1X4

Dangerous Goods Directorate
Transport Canada
Canada Building
Ottawa, Ontario
K1A 0N5

13. Type of Report and Period Covered
Technical

14. Sponsoring Agency Code

15. Supplementary Notes

16. TDC Project Officer
D.W. Dibble

17. Abstract

A prototype computer program has been assembled which is capable of simulating a derailment accident involving up to a maximum of forty rail cars, of which up to eight can be tank cars containing dangerous commodities (chlorine, propane, hexane). The computer program is made up of a number of submodels representing major derailment events, including derailment mechanics, impact induced ruptures, liquid spills, gaseous releases, plume and puff dispersion, fires, fire impingement on tank cars, and explosions.

It is believed that this study represents the first attempt at simultaneously representing all the significant events in a derailment accident. The program developed here is considered to be a prototype for demonstration purposes. As an example, the Mississauga derailment has been simulated.

18. Key Words
Simulation, rail, hazardous, computer, derailment, accident, materials

19. Distribution Statement
Limited copies available from the Transportation Development Centre.

20. Security Classification (of this report)
unclassified

21. Security Classification (of this page)
unclassified

22. No. of Pages

23. Price

Canada

Transports Canada Transport Canada
Développement Development

FORMULE DOCUMENTAIRE DU RAPPORT

1. Nº du rapport Transports Canada
TP 9254E

2. Nº de l'étude — CDT
6835/36/37

3. Nº de catalogue du destinataire

4. Titre et sous-titre
Development of a Derailment Accident Computer Simulation Model

5. Date du rapport
Juin 1988

6. Nº du rapport de l'organisme exécutant

7. Auteur(s)
A.J. Coppens, J.D.E. Wong, A. Bibby, A.M. Birk, R.J. Anderson

8. Nº de dossier Transports Canada
D1465-503

9. Nom et adresse de l'organisme exécutant
W.R. Davis Engineering Ltd.
1260 Old Innes Road
Ottawa (Ontario)
K1B 3V3

10. Nº de dossier — ASC
48SZ.T8200-6-6550

11. Nº de contrat — ASC ou Transports Canada
T8200-6-6550/01-SZ

12. Nom et adresse de l'organisme parrain
Centre de développement des transports(CDT)
200 ouest, boul. René-Lévesque
Suite 601, tour ouest
Montréal (Québec)
H2Z 1X4

Conseil des marchandises dangereuses, Transports Canada
Edifice Canada
Ottawa (Ontario)
K1A 0N5

13. Genre de rapport et période visée
Technique

14. Code de l'organisme parrain

15. Remarques additionnelles

16. Agent de projet — CDT
D.W. Dibble

17. Résumé

Ce rapport traite d'un programme informatique réalisé en prototype et destiné a simuler le déraillement d'un train pouvant comporter jusqu'à 40 wagons, dont huit au maximum peuvent être des wagons-citernes transportant des marchandises dangereuses: chlore, propane ou hexane. Ce programme englobe divers modules permettant de modéliser les phénomènes les plus importants en jeu: cinématique du déraillement, ruptures mécaniques, déversements de liquides, émissions de gaz, dispersion du panache et du souffle, incendies, effet des jets de gaz enflammés et explosions.

Il s'agirait là de la première tentative de prendre en compte simultanément les principaux facteurs en jeu dans un déraillement. Le modèle est encore à l'état de prototype réalisé aux fins d'expérimentations. Le déraillement survenu à Mississauga a été simulé, à titre d'exemple.

18. Mots-clés
Simulation, chemins de fer, marchandises dangereuses, ordinateur, déraillement, accident.

19. Diffusion
Le Centre de développement des transports dispose d'un nombre limité d'exemplaires.

20. Classification de sécurité (de ce rapport)
non classifié

21. Classification de sécurité (de cette page)
non classifiée

22. Nombre de pages

23. Prix

CDT/TDC 79-005
Rev 86

Canada

SUMMARY

This project was conducted by Davis Engineering Limited, in cooperation with A.M. Birk Engineering, for the Transportation Development Centre and Dangerous Goods Directorate of Transport Canada. The project involved the development of a prototype computer program, called DERACS, for the purpose of simulating a train derailment accident involving the release of hazardous commodities. The project included the following major activities:

- A literature search to find viable computer models or documented models that could be used in an overall derailment accident simulator.
- The design of a preprocessor for input of derailment initial conditions and surroundings data.
- The design of a postprocessor for displaying the results of the derailment simulation.
- The design and development of an integrated derailment simulation software package for carrying out the time integration of the derailment consequences.
- The debugging, testing and documentation of the program.

The parameters considered were limited to the following, because of the potentially vast scope of simulating a derailment accident.

a. Vehicle Types
    - Rail (tank cars)

b. Accident Types
    - Derailments

c. Commodities
    - Propane
    - Chlorine
    - Hexane

d. Accident Consequences
    - Pool or torch fire impingement of vehicles
    - Rupture
    - Explosions
    - Gaseous and liquid releases

e. External Influences
    - Ambient temperature
    - Wind speed and direction
    - Topography (in surrounding display only)

f. Model Outputs

- Time integration of accidents events, including
    - Vehicle placement
    - Ruptures due to impact
    - Fires
    - Thermal Ruptures
    - Explosions
    - Releases
    - Area of release influence

Based on the defined scope, a literature search was conducted for the following submodels for use in the overall accident simulator:

- Derailment kinematics
- Derailment impact/rupture analysis
- Spill/release analysis
- Fire analysis
- Thermal rupture analysis
- BLEVE/explosion analysis
- Plume dispersion analysis

Models for the above were acquired and modified, or developed for use in the simulation.

DERACS was structured into three main parts including the pre-processor, simulation software and post-processor.

The pre-processor is used to input data required for the simulation as follows:

- Surroundings map in which the derailment is to take place, i.e. bodies of water, forests, roads, railways, buildings and elevation contour lines
- Reading in all information necessary for the simulation software, including environmental conditions, train consists, and point location of the derailment, if necessary.

The pre-processor then generates a number of files used by subsequent routines. The simulation software carries out the various calculations to predict the consequences of derailment. The simulation software includes all the necessary mathematical models of the various processes that will take place during the derailment. There are six main areas in the accident simulation software: derailment mechanics, liquid and gaseous releases, fire impingement, tank car thermal model, explosions, and plume and puff dispersion. This part of the program then sends the data to output files for later processing.

The post-processor takes all the files generated by the pre-processor and the simulation software and summarizes them in graphical and text form for the user. The post-processor gives a plan view of the surroundings and shows placement of the vehicles after the derailment and the various events taking place such as spills, gaseous releases, fires, thermal ruptures and explosions. The post-processor also gives a text summary of the accident simulation, which includes what has happened, when it happened and what areas of the map were affected.

The computer program, developed for operation on an IBM AT or compatible computer, is a prototype software package for simulating derailment accidents involving a maximum of forty rail cars of which up to eight can be tank cars.

For verification purposes, sample simulations were conducted including the Mississauga derailment. Agreement between the predicted and actual results was good in the case of the Mississauga derailment. The number of cars that derailed in both cases were simular, as well as the final location of the rail cars after the derailment. The consequences of the accident, including spills, ruptures, explosions and fires, were also well represented by the simulation program. The program has proved to be capable of simulating a derailment accident with some degree of realism. The principal elements of the simulation are accounted for, and are based on cause and effect as opposed to statistics. However, the prototype program does have weaknesses. These weaknesses are present because of certain assumptions, which were necessary to limit computer run time and memory demands.

To develop DERACS into a comprehensive software package for simulating rail and highway accidents involving tankers carrying dangerous commodities, a number of enhancements are required. These include taking into account more dangerous commodities, terrain effects, tank rocketing and the thermal reaction between chlorine and steel; improving rupture prediction capabilities and derailment mechanics calculations; and, upgrading the plume and puff dispersion model.

The DERACS program should also be studied for other potential applications, including risk analysis and assessment, response training, accident reconstruction and forecasting. This could be done by looking at how the nuclear power and aircraft industries have used computer simulations to improve safety in their industries. The results of the application study would identify the areas where DERACS should be further developed. The program should also be developed for application on more powerful computers so that some of the limiting factors (number of tanks, number of commodities, etc.) can be eliminated.

SOMMAIRE

Menée par la Davis Engineering Limited en collaboration avec la A.M. Birk Engineering, la recherche a été commanditée par le Centre de développement des transports et la Direction générale de transport des marchandises dangereuses, deux organismes de Transports Canada. Elle a visé l'étude et la création d'un programme informatique prototype appelé DERACS, capable de simuler un déraillement entraînant le déversement ou l'émission de produits dangereux. La recherche a nécessité les travaux suivants:

- Recherche documentaire sur des modèles existants ou sur papier pouvant servir à la simulation.

- Etude d'un préprocesseur permettant de modéliser les circonstances entourant le déraillement ainsi que les détails topographiques.

- Etude d'un postprocesseur permettant d'afficher les résultats de la simulation.

- Etude et mise au point d'un progiciel permettant l'intégration dans le temps des événements successifs.

- Déverminage, essai et compte rendu du programme.

Quant aux paramètres pris en compte, ils ont été restreints à ceux qui sont indiqués ci-dessous, étant donné qu'il n'était pas possible de dégager tous les paramètres en jeu, trop nombreux:

a. Catégorie de matériel roulant
   - wagons-citernes

b. Catégorie d'accident
   - déraillement

c. Marchandises transportées
   - Propane
   - Chlore
   - Hexane

d. Conséquences
   - Flammes enveloppantes ou jet de gaz enflammé
   - Rupture des parois
   - Explosion
   - Déversement/émission de liquides et de gaz

e. Influences externes
   - Température ambiante
   - Vitesse et direction du vent
   - Topographie du secteur immédiat visualisé

f. Sorties informatiques
- Intégration dans le temps des conséquences du déraillement:
  - position des wagons
  - ruptures mécaniques
  - incendies
  - ruptures thermiques
  - explosions
  - déversements/émissions
  - zone d'influence du déraillement

A partir de ces paramètres, une recherche documentaire a été faite pour trouver des modules pouvant servir à simuler les aspects suivants:

- cinématique d'un déraillement,
- analyse des impacts et des ruptures suite à un déraillement,
- analyse des déversements/émissions,
- analyse du comportement des flammes,
- analyse des ruptures thermiques,
- analyse des explosions dues à la dilatation des vapeurs d'un liquide en ébullition,
- analyse de la dispersion du panache.

Les modules acquis ont été modifés ou adaptés pour servir à l'usage prévu.

Le programme DERACS est constitué de trois grands ensembles qui sont le préprocesseur, le logiciel de simulation et le postprocesseur.

Le préprocesseur permet d'appliquer au modèle des valeurs d'entrée concernant:

- la topographie du lieu du déraillement: plans d'eau, boisés, routes, voies ferrées, bâtiments et courbes de niveau;
- les informations requises par le logiciel de simulation: conditions externes, composition du train et, au besoin, repérage du lieu du déraillement.

Le préprocesseur produit un certain nombre de fichiers qui serviront au déroulement du programme. Le logiciel de simulation fait les calculs nécessaires permettant de prévoir les conséquences du déraillement, grâce aux équations mathématiques en mémoire représentant les divers phénomènes impliqués dans le déraillement. Les phénomènes pris en compte ont été les suivants: cinématique du déraillement, déversements de liquides et émissions de gaz, effets d'un jet de gaz enflammé, flux thermiques dans un wagon-citerne, explosions, dispersion du panache et du souffle. Le résultat des calculs est transmis aux fichiers sortie en vue de leur traitement ultérieur.

Le postprocesseur se saisit des données produites durant les étapes précédentes, les réduit et les affiche sous forme infographique accompagnée de texte; donne une vue en plan du lieu et montre la position des wagons après le déraillement. Il présente également les conséquences de ce dernier et leur effets: déversements, jets de gaz, incendies, ruptures thermiques et explosions. Enfin, il fait un compte rendu de la simulation dans tous ses détails, déroulement des événements dans le temps et zone d'influence de l'accident.

Le programme infromatique conçu pour fonctionner sur un AT d'IBM ou tout autre matéeriel compatible, est le prototype d'un progiciel devant permettre de simuler le déraillement d'un train comportant jusqu'à 40 wagons, dont huit aux maximum sont des wagons-citernes.

Pour vérifier la qualité des résultats, le modèle a été soumis à des essais de simulation, à partir d'accidents connus, celui de Mississauga, par exemple. Il a donné une bonne approximation des conséquences de ce dernier, comme le nombre exact de wagons ayant déraillé ainsi que leur position relative. Les conséquences de ce déraillement: déversements, ruptures, explosions et incendies, ont également été bien représentées. Ainsi, le modèle s'est montré apte à simuler un déraillement avec un certain degré de réalisme. Les principaux éléments du déraillement simulé sont corroborés par les faits et sont fondés sur des relations analytiques vérifiées et non pas sur des extrapolations statistiques. Par contre, le modèle présente des faiblesses dues au fait qu'il a fallu au départ retenir certaines hypothèses destinées à écourter les phases d'exécution et à restreindre la capacité de mémoire à mettre en oeuvre.

Des perfectionnements seront nécessaires pour amener le modèle à un stade de développement tel qu'il pourra prendre en charge la totalité des opérations de simulation d'accidents ferroviaires et routiers, durant le transport de marchandises dangereuses, liquides ou gazeuses. Parmi ces perfectionnements citons: la prise en compte d'une gamme plus vaste de marchandises dangereuses et des détails topographiques du lieu du déraillement, effet brisant de l'explosion et les réactions thermiques entre le chlore et l'acier; augmenter la capacité de prévision et de calcul des ruptures dues à des causes mécaniques, thermiques ou cinématiques et, enfin, faire monter en puissance le modèle de dispersion du panache et du souffle.

Mais le modèle devra être envisagé sous l'angle d'autres applications futures, comme l'analyse et l'évaluation des risques, la formation des agents d'intervention, la reconstitution et la prévision des accidents. Cela pourrait se faire en s'inspirant de ce que l,industrie nucléaire et les constructeurs d'avions ont fait dans le domaine de la simulation dans l'intérêt de la sécurité, et en déterminant les nouvelles applications que le modèle pourra prendre en charge. Enfin, il faudra prévoir son utilisation sur des calculatrices encore plus puissantes de manière à pouvoir éliminer les contraintes limitant les facteurs pris en compte, comme le nombre de wagons, celui de marchandises transportées, etc.

TABLE OF CONTENTS

1. INTRODUCTION 1
   1.1 Background 1
   1.2 Objectives and Scope 1

2. PROJECT OVERVIEW 3
   2.1 Literature Search 3
   2.2 Software Design 5

3. SIMULATION SOFTWARE OVERVIEW 6
   3.1 DERACS Organization 6
   3.2 Pre-Processor 6
   3.3 Simulation Software 8
   3.4 Post-Processor 8

4. INPUT PREPROCESSOR 9
   4.1 Introduction 9
   4.2 Description 9
   4.3 Description of FORTRAN Routines 10

5. DERAILMENT SIMULATION SOFTWARE 14
   5.1 DERAIL 14
   5.2 DERACS 14
      5.2.1 DERACS Main Program 14
   5.3 Primary Subroutines 37

6. POST-PROCESSOR 46
   6.1 Introduction 46
   6.2 Description of Subroutines 47

7. SAMPLE SIMULATION 49
   7.1 The Mississauga Derailment 49
   7.2 DERACS Simulation Input 52
   7.3 Sample Simulation Results 55
      7.3.1 Simulation Case 1 - Derailment Mechanics Used to Predict Vehicle Placement 55
      7.3.2 Simulation Case 2 - Input of Vehicle Placement and Simulation of Resulting Consequences 59
   7.4 Discussion 59

8. FUTURE DEVELOPMENT 71
   8.1 Program Weaknesses 71
      8.1.1 General Weaknesses 71
      8.1.2 Derailment Mechanics 73
      8.1.3 Tanker Fire Impingement 74
      8.1.4 Releases, Plumes, Puffs 74
      8.1.5 Explosions 75
   8.2 Program Strengths and Potential Applications 75

9. CONCLUSIONS AND RECOMMENDATIONS 77
   9.1 Conclusions 77
   9.2 Recommendations 77

10. REFERENCES 78

TABLE OF CONTENTS (continued)

APPENDICES

APPENDIX A Interim Report
APPENDIX B Program Listing
APPENDIX C User Guide
APPENDIX D Simulation Software Technical Descriptions
APPENDIX D.1 Program DERAIL
APPENDIX D.2 Subroutine BLAST
APPENDIX D.3 Subroutine BLEVE
APPENDIX D.4 Subroutine CONREL
APPENDIX D.5 Subroutine FIRBAL
APPENDIX D.6 Subroutine FIRIMP
APPENDIX D.7 Subroutine FLARE
APPENDIX D.8 Subroutine PFIRE
APPENDIX D.9 Subroutine SPILL
APPENDIX D.10 Subroutine TORIMP
APPENDIX D.11 Subroutine PLUME AND PUFF

LIST OF FIGURES

Figure 3.1 Overall DERACS Organization 7
Figure 5.1 Flow Diagram for Simulation Software 16
Figure 7.1 Mississauga Derailment Site (From Reference [5] 50
Figure 7.2 Approximate Map of Derailment Site (From Reference [5] 53
Figure 7.3 Input Vehicle Placement and Location of Ruptures 56
Figure 7.4a Prediced Vehicle Placement and Rupture Locations (Zoom Level 2) 57
Figure 7.4b Prediced Vehicle Placement and Rupture Locations (Zoom Level 3) 58
Figure 7.5 Case 1 Simulation Results At 0.3 Minutes 60
Figure 7.6 Case 1 Simulation Results At 3.3 Minutes 61
Figure 7.7 Case 1 Simulation Results At 3.3 Minutes 62
Figure 7.8 Case 1 Simulation Results At 9.7 Minutes 63
Figure 7.9 Case 1 Simulation Results At 22 Minutes 64
Figure 7.10 Case 2 Simulation Results At 0.3 Minutes 65
Figure 7.11 Case 2 Simulation Results At 0.3 Minutes 66
Figure 7.12 Case 2 Simulation Results At 1.7 Minutes 67
Figure 7.13 Case 2 Simulation Results At 5.7 Minutes 68
Figure 7.14 Case 2 Simulation Results At 24.3 Minutes 69

LIST OF TABLES

Table 2.1 Recommended Models to be used in Simulation Software Development (From Interim Report) 4
Table 5.1 Summary of Primary Subroutines called by DERACS 37
Table 5.2 Summary of Data Printed to File By Dump 39
Table 7.1 Summary of Derailed Cars and Their Contents 51
Table 7.2 Summary of Consist Used in DERACS Simulation of Mississauga Derailment 54
Table 8.1 Summary of DERACS Simulation Weaknesses 72

1. INTRODUCTION

This report was prepared by W.R. Davis Engineering Limited in cooperation with its subcontractor, A.M. Birk Engineering for the Transportation Development Centre and Dangerous Goods Directorate of Transport Canada. The work was conducted under DSS contract number 48SZ.T8200-6-6500/01-SZ. The project involved the development of a Derailment Accident Computer Model.

1.1 Background

Over the last two decades, there has been a considerable amount of work carried out to develop modelling or analysis techniques to quantify the consequences of rail or highway accidents involving hazardous commodities. These developments have been in many areas, including: accident mechanics (both from vehicle motion and vehicle fire impingement standpoints), release rates and dispersion of dangerous goods, and fire and explosion effects on neighboring targets. However, in most cases these models or techniques stood alone with little interaction with the mechanisms that occurred in parallel or series with them.

The present project was an attempt to integrate some of these available models into an overall derailment accident computer simulation. This project was to develop a prototype version of such a simulation for the purposes of evaluating the usefulness of such a simulation tool.

The project involved:

a. A literature search to find available computer models or documented models that could be used in an overall derailment accident simulator.

b. The design of a pre-processor for input of derailment initial conditions and surroundings data.

c. The design of a post-processor for displaying the results of the derailment simulation.

d. The design of an integrated derailment simulation software package for carrying out the time integration of the derailment consequences.

e. The development of the above software and its debugging, testing and documentation.

1.2 Objectives and Scope

As set out in the original proposal to carry out this work, the project objective was to develop a prototype accident computer model capable of simulating, from start to finish, an accident involving the release of hazardous commodities. The scope was limited to the development of a prototype software package.

Because of the potentially vast scope of simulating a derailment accident, it was necessary to define the capabilities of the derailment simulator.

The computer program was to include appropriate preprocessor software for setting up the simulation conditions, and a postprocessor for summarizing the simulation results.

As set out in the original proposal (with some additions) to do the study, the scope was limited to:

a. Vehicle types
   - Rail (tank cars)

b. Accident Types
   - Derailments

c. Commodities
   - Propane
   - Chlorine
   - Hexane (added)

d. Accident Consequences
   - Pool or torch fire impingement of vehicles
   - Rupture
   - Explosions
   - Gaseous and liquid releases

e. External Influences
   - Ambient temperature
   - Wind speed and direction
   - Topography (in surroundings, display only)

f. Model Outputs
   - Time integration of accident events, including:
     - Vehicle placement
     - Ruptures due to impact
     - Fires
     - Thermal Ruptures
     - Explosions
     - Releases
     - Area of release influence

2. PROJECT OVERVIEW

The project has resulted in the development of a prototype computer program for simulating a derailment accident. Prototype in the present project means a limited scope program that is not intended for release. The program has not been validated and has significant weaknesses, as will be discussed later in the report.

The program has been given the name DERACS (DERailment ACcident Simulator). This name will be used extensively throughout this report.

The project involved the following major tasks:

a. Literature search and review

b. Software design

c. Software development

d. Sample simulation and final report

The results of a. and b. above were summarized in an interim report and can be found in Appendix A.

Tasks c. and d. are summarized in this report.

Since the preparation of the interim report, a number of changes have occurred with respect to the model development and the software design. These changes are summarized in the following sections.

2.1 Literature Search

The overall results of the literature search are described in the Interim Report (Appendix A). Since the preparation of this report, a number of changes have taken place with regard to the availability or applicability of software or models recommended for use in the Interim Report. Table 2.1 presents the recommended models as given in the Interim Report.

Since then the following major changes have occurred:

a. Derailment Mechanics

The Yang and Manos software has not been used since it was decided that the effects of uncoupling and tank roll should be included. For this it was necessary to develop new models. Details of the derailment mechanics software can be found in Appendix D.

Table 2.1 Recommended Models to be Used in Simulation Software Development (From Interim Report)

SUBMODEL	SOURCE	STATUS
Derailment Mechanics	- Yang and Manos in house	- To be further developed to allow for uncoupling and roll to facilitate possibility of side impacts
Impact Rupture	- Yin - Phillips - to be coded	- To be further developed to include side impact ruptures - Impact model in the form of equations to be coded
Spill/ Release	- Fingas - "LEAKRATE" in house	- Minor modifications for integration
Fire	- Crocker - "HOTPLT" in house - Birk	- Crocker analysis to be coded - Minor modifications for integration - Some fire effects to be handled by tank thermal model
Thermal Rupture	- Birk - "TANKCAR" in house	- Modifications to simplify for improved speed
BLEVE/ Explosion	- Reid - Leuschen and Birk, in house	- Reid analysis to be coded - Minor modifications for integration with other models
Plume Dispersion models	- Morgan - "SLAB" procured - Peterson in house	- Minor modifications for integration - Puff model to be integrated other to be

b. Plume and Puff Dispersion

The recommended time dependent plume model SLAB is no longer available and as a result the plume and puff modelling approach has been revised. In DERACS, plumes and puffs are modelled using a box type model as described by Merony et al. Details of these models can be found in Appendix D.

2.2 Software Design

The basic software design is described in the Interim Report (Appendix A). Since then the pre and post processor has been enhanced to allow user input through a pointing device (mouse). This change makes input of the surroundings and train data a much more efficient process. The details of the pre and post processors are given in Chapters 4 and 6 of this report.

3. SIMULATION SOFTWARE OVERVIEW

This chapter presents an overview of the DERACS software. The program can be broken down into three main parts: the pre-processor, the simulation software, and the post-processor.
These three parts will be described separately following a basic introduction to the DERACS organization.

3.1 DERACS Organization

As stated above the DERACS program consists of three main parts: the pre-processor, the simulation software and the post-processor.

The pre-processor is used to input scenery and initial conditions data required by the simulation software. The pre-processor then generates a number of files that are used by subsequent routines.

The simulation software carries out the various calculations to predict the consequences of a derailment. This part of the program then sends data to output files for later processing.

The post-processor takes all the files generated by the pre-processor and the simulation software and summarizes them in graphical and text form for the user. Figure 3.1 presents a block diagram summary of the overall program organization.

3.2 Pre-Processor

The pre-processor is used to create the surroundings in which the derailment is to take place. It is a user friendly routine that allows the user with the aid of a mouse to draw a map including such things as: rivers and lakes, forests, roads of different sizes, railways tracks either single or double, buildings, property boundaries, and elevation contour lines. The DERACS pre-processor uses an appropriate colour scheme so that the resulting map of the surroundings is easy to interpret. The total map area is 10 kilometers by 10 kilometers and the user can input part or all of this area when creating the map.

In addition the pre-processor has the function of reading in all the other necessary information for the simulation, including environmental conditions, such as wind speed and direction, ambient temperature and atmospheric pressure. The pre-processor is also used to select a train consist including locomotives, tank cars, and dummy cars. Depending on the options selected the pre-processor may also read in the point location of the derailment and the car number, truck number and axle number, which initiates the derailment. The train speed and direction is also read in during the pre-processor session.

The user has the option of overriding the derailment mechanics simulation portion of the overall simulation and if he/she chooses this option the user must specify the final placement and orientation of vehicles after the derailment, which cars are ruptured and what kind of ruptures have occurred. Further details of the input pre-processor will be described in Chapter 4.

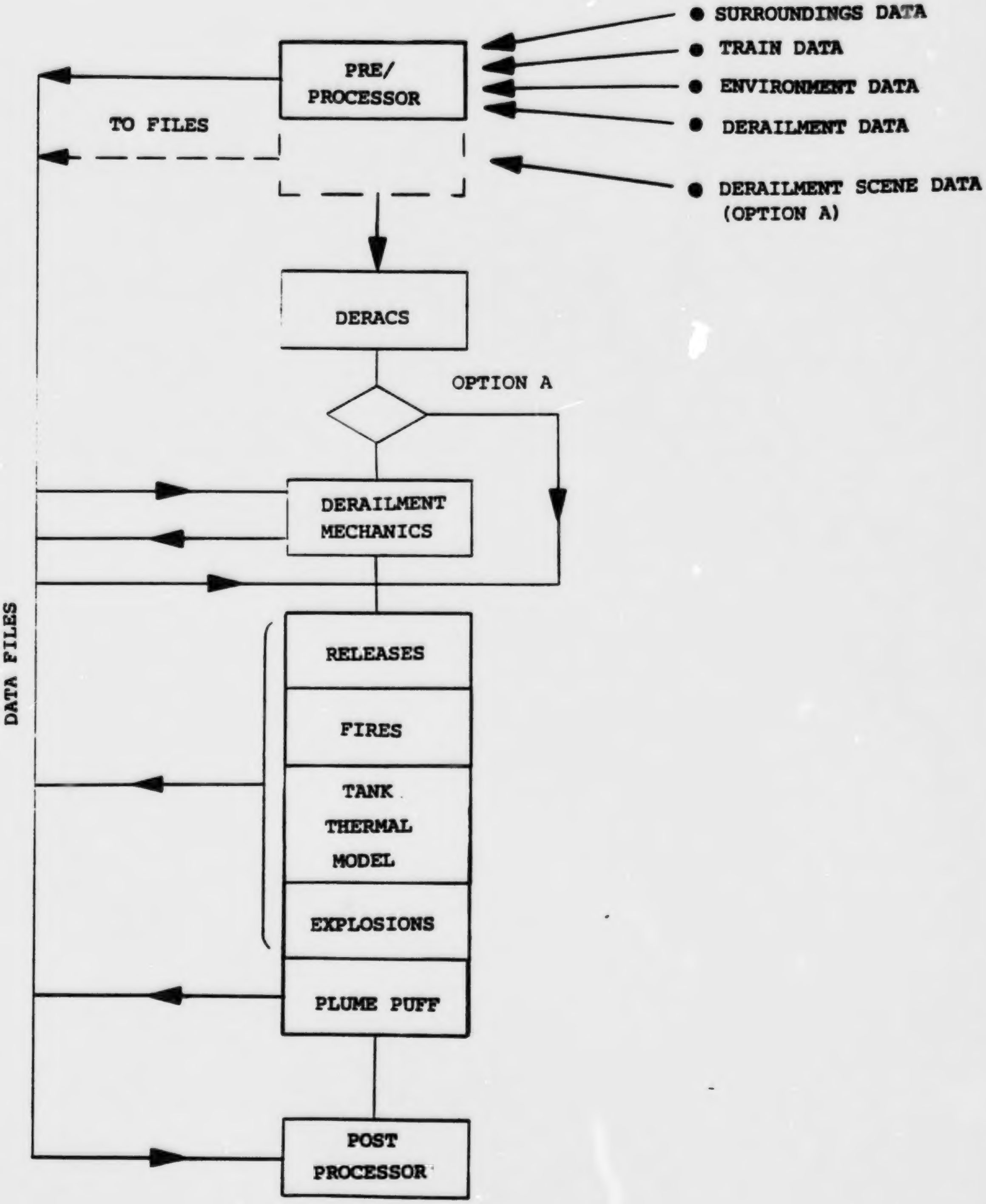

SURROUNDINGS DATA
TRAIN DATA
ENVIRONMENT DATA
DERAILMENT DATA
DERAILMENT SCENE DATA (OPTION A)
PRE/ PROCESSOR
TO FILES
DERACS
OPTION A
DERAILMENT MECHANICS
DATA FILES
RELEASES
FIRES
TANK THERMAL MODEL
EXPLOSIONS
PLUME PUFF
POST PROCESSOR

Figure 3.1 **Overall DERACS Organization**

3.3 Simulation Software

The simulation software includes all the necessary mathematical models of the various processes that will take place during the derailment. There are six main areas in the accident simulation software and these include: derailment mechanics, liquid and gaseous releases, fire impingement, tank car thermal model, explosions, and plume and puff dispersion.

The simulation software consists of one or more subroutines which are used to represent each one of these processes. The main program of the simulation software called DERACS calls these routines in a logical fashion to carry out the time integration of the derailment accident simulation. These various subroutines will be described in more detail in Chapter 5.

3.4 Post-Processor

The final part of the overall DERACS software is one of the most important because it presents the results of the simulation to the user in a easily understandable fashion. The post-processor is used to read files that contain the results of the simulation and presents this information in graphical or text form to the user at the end of the simulation. By using the post-processor the user can step through the simulation process and look at various instants in time during the simulation to see how the accident progressed.

The post-processor gives a plan view of the surroundings and shows placement of the vehicles after the derailment and the various events that are taking place such as spills, gaseous releases, fires, thermal ruptures, explosions, etc. The user can select an instant in time and print out a snapshot of the derailment scene. The post-processor also gives a text summary of the accident simulation which includes what has happened, when it happened and what areas of the map were affected. Further details of the post-processor are given in Chapter 6.

4. INPUT PREPROCESSOR

4.1 Introduction

The preprocessor design was presented in the Interim Report (Appendix A) has been improved. The original design was complete but was lacking in understanding and user friendliness. The main areas of enhancement lie in data input and presentation. The preprocessor uses a mouse for selecting commands and locating points on the screen. The preprocessor also allows visual placement of objects, constant screen updating and an UNDO option which allows you to delete last input.

4.2 Description

The preprocessor screen is broken up into 3 areas. They are: menu, drawing and status area. The menu area contains the functions or commands which the user selects as data is entered. The drawing area is where the map is drawn and the status area shows continuous update of necessary information.

The input logic for the preprocessor is as follows:

a) Train Consist Data

- number of cars
- type of cars
- thermal protection
- commodities
- fill level
- initial temperature
- train speed
- car, truck, axle that initiates derailment

b) Track

- derailment section - double, single
- other - double, single

c) Ground Surrounding Data

- buildings (square, rectangles)
- boundaries
- water and forest
- roads - small, medium, large
- contours indicating levels of elevation

d) Atmospheric Conditions

- wind speed
- wind direction
- air temperature
- atmospheric pressure

e) Derailment Data

- position and angle of vehicle
- roll angle of vehicle
- tank rupture status
- description of rupture.

The method of input for section a) is done via the keyboard. Each prompt appears in a separate window and is illustrated in Figure C4.1 of the USER GUIDE (Appendix C).

Section b) is input using a combination of mouse and keyboard. Derailment track requires more input than the visual track, as more detail on track position is necessary for calculations on vehicle placement.

Section c) is entered almost entirely using the mouse. The mouse is used to locate positions and dimensions of roads, buildings, etc. The construction of the grounds or map can be done at various levels of zoom allowing accuracy of up to 1 meter on a 10,000 X 10,000 meter area.

Atmosphere data, Section d), is input using the keyboard. The prompts appear in windows and are illustrated in Figure C4.4 of the USER GUIDE.

Derailment Data, Section e), is also input using the keyboard. All angles and distances are input with reference to the Derailment Coordinate System (D.C.S.). For an illustration of the D.C.S., refer to Figure C4.2 in the USER GUIDE.

4.3 Description of FORTRAN Routines

This section describes the major routines which make up the preprocessor. The format will be subroutine name, description of subroutine and description of important variables. Because variables are used by many routines, they will only be described once.

DRW - This is the main loop of the preprocessor. In this routine, the mouse is polled for its location and condition. If the mouse button is pressed, a flag is set and the appropriate routine is called according to the current mouse position in the menu selection. Upon completion of the action, the status area is updated and the control is returned to the main loop.

ICNUM	- (I) number of icon selected
EXITT	- (L) exit flag
SWITCH	- (I) mouse condition
XR, YR	- (R) X and Y coordinates of mouse (cursor)
CLICK	- (L) flag for mouse condition. Set to true if left mouse button has been pressed

SUBZOM - This function controls the zooming. It allows the world coordinates to be changed to suit user request. The zoom level dimensions are as follows:

10000 m^2
1000 m^2
200 m^2

TYPE - (I) This variable is set to 1 if zoom-in has been requested and set to Z if zoom-out has been requested.

ILEVEL - (I) current zoom level

XB, YB,
XT, YT - (R) Current world coordinates

SUBRDR - This function redraws the screen reflecting any changes made to the surroundings or changes made to the world coordinates. Items are redrawn on the screen in a fixed order, regardless of how they were input. The order is as follows:

1. Water and forest
2. Roads, contours lines, visual track and boundaries
3. Buildings
4. Derailment track
5. Cars - locomotive, box, tank
6. Events - blast, fire, puff, plumes, flares, spills

ITRACK - (I) set to 2 if derailment track exists

NCARS - (I) number of cars

UNDO - This function decrements the display list counter and, by doing so, deletes the last entry drawn on the screen. The screen is redrawn before returning.

KOS - display list counter

EXITD - This function checks to see if all data has been entered. Data such as train, atmosphere, or track must be input for the simulation to continue. When the data has been entered, the output files are opened.

EXITT - (L) this flag is set to .true. before returning.

KEY - This function displays the Key Page and then polls the mouse condition. When either button is pressed, the page is switched and the routine exits.

DRAW1 - This function is used to enter train consist data. Windows are drawn in the menu area, which contains prompts. As the train is being entered, the cars appear from the left side of the screen. When the train is entered, the user is prompted for derailment data.

The maximum number of tank cars that can be entered is 8 and the total number of cars is 40. When either of these limits are exceeded, the function MAXCAR is called, which displays a message warning that the number of cars has been exceeded. After each car has been entered, the function DRC draws a new car on the screen and the status area is updated.

TRAK - This function is used to enter track data. The three types of track are Derailment, Joined and Other. Each of these three types can be either double or single; therefore, that is the first prompt which is shown. The next prompt is track type. Track types Joined and Other are similar, as they are both visual track and, therefore, only endpoints of each section are stored. Derailment track can be either curved or tangent. Tangent track is really just a curved track with a very large radius of curvature but it is treated as a special case. If tangent track is chosen, the two endpoints are input by mouse location and the function DTANGT is called. DTANGT allows the user to locate the derailment point by making the cursor move along the section of tangent track. If curved track is chosen, the centre of curvature is entered via the mouse. The start and end angles are then input by keyboard. At this point, the two endpoints are calculated and stored for use by the Joined function. To locate the point of derailment on curved track, the function DERTRK is used. In both DTANGT and DERTRK, angles are calculated and later used to translate the derailment coordinate system into world coordinates.

ITRACK - (I) Flag set to 2 if derailment track is entered

IWIDTH - (I) Width of derailment track

DIRECT - (R) Direction of train motion

BETA, THETA
UANGLE - (R) Angles used to translate derailment coordinate system

D1X, D1Y,
D2X, D2Y - (R) Two end points of derailment track

RCURV - (R) Radius of curvature

SUBDER - This function is used to enter derailment mechanics data. This data is entered entirely by keyboard; therefore, the graphics screen is cleared and FORTRAN "write" statements are used as prompts. For details on the prompts, refer to the secion on Derailment Mechanics. After all data is input, the graphics are refreshed and the cars appear at the input coordinates.

DERAIL - Set to 2 after derailment mechanics has been entered.

DRAW10 - This function is used to draw buildings. Two points of a square are input using the mouse and a filled square is drawn.

DRAW11 - This function is used to draw all line functions. The method of drawing is outlined in the user manual.

TYPE - (I) Type of line to be drawn

DUMP - This function writes the picture data and DERACS data to the appropriate files. The file names and format is fixed so the other DERACS modules can interpret them.

SUBCAR - This function is used to draw the cars at their designated positions. It should be noted that the translated position is calculated in the function CALCAR when the derailment mechanics are manually entered. The cars are rotated here and the markings are also calculated.

RCRS - This function is used to read DERAC data files and a picture file. The files are predefined and have a fixed format.

RCARS - (R) This is an array variable containing car data

NCARS - (I) Number of cars

UWIND - (R) Wind direction in radius from 3:00 c.c.w.

VANGL - (R) Rotation angle.

PATM - (R) Atmospheric pressure

TATM - (R) Atmospheric temperature

RCURV - (R) Radius of curvature

ICARDR - (I) Derailment leader

ITRUCK - (I) Truck #

IAXLE - (I) Axle #

DX, DY - (R) Derailment point

WSPEED - (R) Wind speed.

5.0 DERAILMENT SIMULATION SOFTWARE

The simulation software consists of two programs: DERAIL.EXE and DERACS.EXE. The DERAIL program is used (if the appropriate option is selected by the user) to predict vehicle placement after a derailment and to predict impact induced ruptures. The DERACS program is used to carry out all the other simulation calculations.

5.1 DERAIL

This program is run if derailment scene data is not available for post derailment vehicle placement and impact rupture.

DERAIL reads in train consist data (number of vehicles, location of tank-cars, train speed, track curvature, and car, truck and axle that initiates derailment). The program then predicts the vehicle motion and resulting impacts.

DERAIL can simulate the derailment mechanics for up to 40 cars. The model considers flat terrain and allows for 4 degrees of freedom (x, y, yaw, roll). Vehicle uncoupling is also accounted for.

During the mechanics simulation, vehicle-vehicle impacts are monitored and probable rupture modes are predicted. Rupture modes presently considered include total loss of containment, and round inward punctures of variable diameter and location along the length of the tank-car (only tank-cars are considered in the impact rupture analysis).

DERAIL then sends data to a file DERAIL.DAT including vehicle coordinates, yaw, roll, and if the vehicle is a tank-car, the data includes rupture type, diameter, height and position on the car. This file then becomes a primary input file for the remaining derailment simulation.

Technical details of the program DERAIL are contained in Appendix D.1.

5.2 DERACS

The DERACS program controls the remaining simulation calculations, including those for spills and releases, fires, tank-fire impingement, explosions, and plume and puff dispersion. Since this program is the heart of the simulation, it will be described in detail.

5.2.1 DERACS Main Program

The heart of the derailment simulation software is the main program called DERACS which calls the various subroutines that contain the sub-models. The following is an overview of this main program. Appendix B contains a hard copy listing of this main program and the subroutines that go along with it.

Figure 5.1 presents a flow diagram for the main program showing the various calls to the subroutines.

The DERACS main program starts with the various declarations of variables, arrays and matrices and the definition of the various common blocks for the program. These declaratio are then followed by a number of calls to subroutines.

The first subroutine to be called is OPFILE and this subroutine is used to open the various files that are used for input and output from the DERACS simulation. This subroutine call is then followed by a number of statements that are used to initialize the simulation time and time step (20 seconds). These statements are then followed by calls to subroutines that initialize the variables and arrays that are specific to the derailment simulation.

The next subroutine to be called is SIMCOM and this subroutine is used to read files CARDATA.DAT and DERAIL.DAT which contain all the data that is needed to perform the derailment simulation. The next call is to subroutine READ which is used to read in the thermodynamic property data for propane, chlorine, and hexane. This data is of course needed to perform the tank thermal calculations.

The next block of the main program is a do-loop which considers each tank car in turn. This block or do-loop is basically the initialization of the simulation where it establishes whether or not the tank was ruptured during the derailment, and if ruptured, the kind of rupture it was and the immediate consequences of the rupture. In this do-loop the first thing that happens is a call to subroutine INIT which initializes all the tank car thermal model variables and arrays for the current tank car. This is followed by an if statement which establishes whether the tank car has been ruptured during the derailment. If the tank has not been ruptured then the program proceeds to the next tank car in the do-loop.

If the current tank is ruptured the next statements increment the counter which tracks the number of tanks that have ruptured and also sets the time of the rupture for that specific tank. The next lines in the code are if statements which establish the commodity of the current car which has been identified as being ruptured. If the ruptured tank contains propane then two possible consequences have been assumed. If the rupture was a total loss of containment of the propane then it is assumed that the contents are expelled and a fire ball results. The flags are set for this current tank car to show that it has resulted in an explosion and that the tank is now empty of its contents. The time of the explosion is also set. Subroutines INSREL, BLEVE, FIRBAL, and BLAST are called in sequence. These subroutines are used to calculate the mass of the material released due to the total loss of containment of the tank, to establish whether a BLEVE is likely, and to calculate the size of the resulting fire ball and its thermal and blast effects.

If it is established that the current tank which contains propane has a round inward puncture then it is assumed that this puncture will result in a continuous release of propane and that this propane will form a flammable cloud that will drift downwind. In this case then the flags are set to show that a plume is associated with this rupture and the time of the release is set. Subroutines CONREL and PLUME are then called to calculate the rate of the continuous release from the puncture and to track the growth of the flammable cloud downwind.

If it was established that the current tank car contained chlorine then different consequences are assumed to occur. If the rupture type for this tank car containing chlorine is established to be a total loss of containment then it is assumed that a puff occurs involving the total mass contained in the tank at the time of rupture. Flags are then set to show that the tank is empty and also the time of the puff is set. Subroutines are called to calculate the mass contained in the tank at the time of rupture and the puff dispersion routines are called to track the progress of the puff for the simulation.

If however, it is determined that the puncture type is a round inward puncture then subroutines will be called to track the continuous release from this tank. Flags are set to show that a plume is emanating from the tank and the time of the plume is set. Subroutines are called to calculate the continuous release rate from the tank and plume routines are called to track the progress of the toxic cloud as it moves downwind.

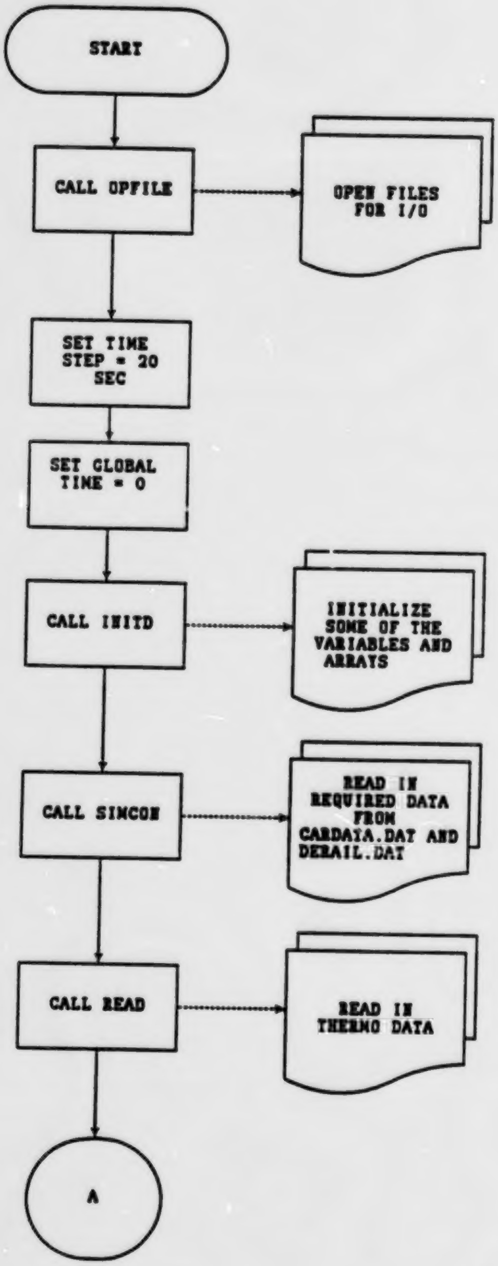

START
CALL OPFILE
OPEN FILES FOR I/O
SET TIME STEP = 20 SEC
SET GLOBAL TIME = 0
CALL INITD
INITIALIZE SOME OF THE VARIABLES AND ARRAYS
CALL SIMCON
READ IN REQUIRED DATA FROM CARDATA.DAT AND DERAIL.DAT
CALL READ
READ IN THERMO DATA
A

Figure 5.1 Flow Diagram for Simulation Software

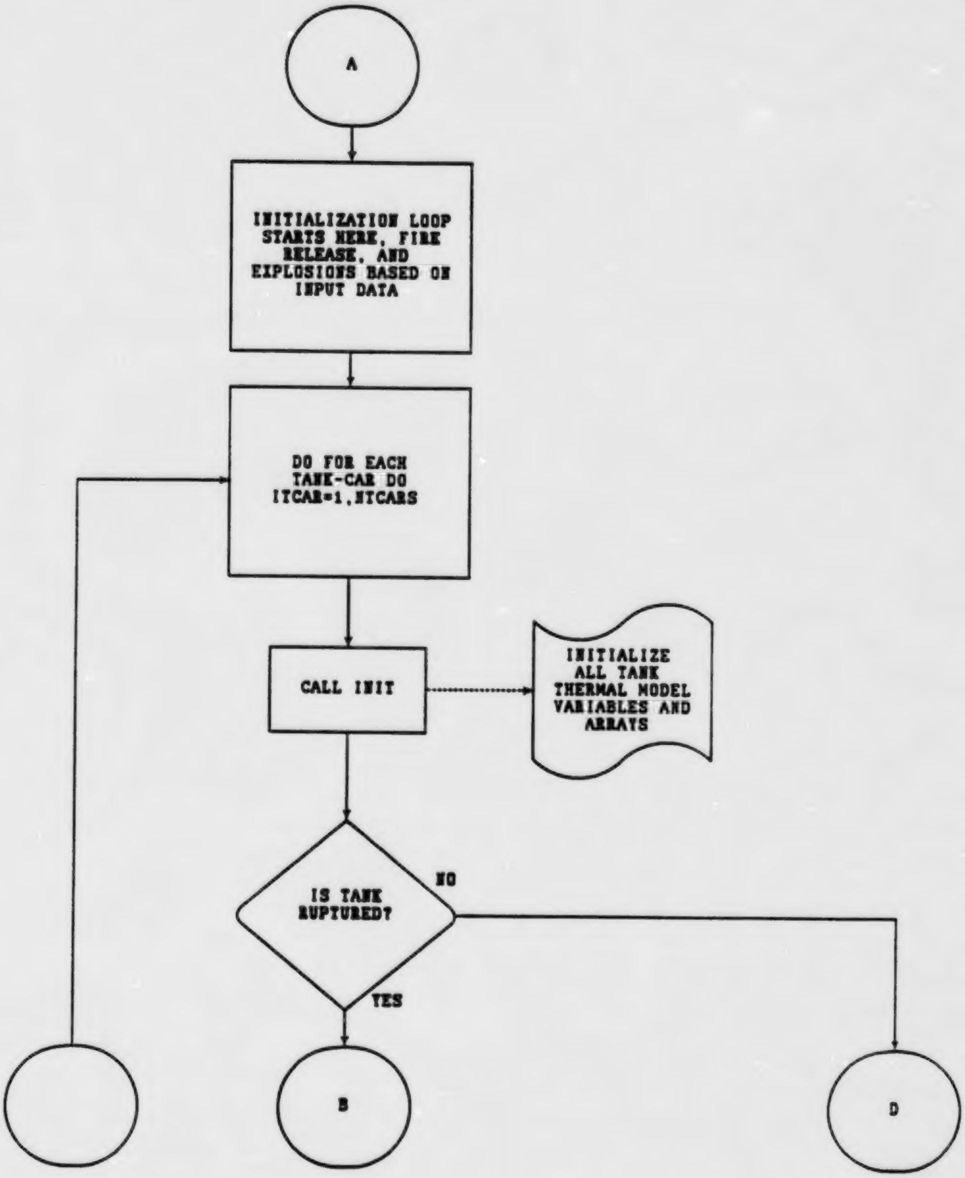

A
INITIALIZATION LOOP STARTS HERE, FIRE RELEASE, AND EXPLOSIONS BASED ON INPUT DATA
DO FOR EACH TANK-CAR DO ITCAR=1,NTCARS
CALL INIT
INITIALIZE ALL TANK THERMAL MODEL VARIABLES AND ARRAYS
IS TANK RUPTURED?
NO
YES
B
D

Figure 5.1 (cont'd) Flow Diagram for Simulation Software

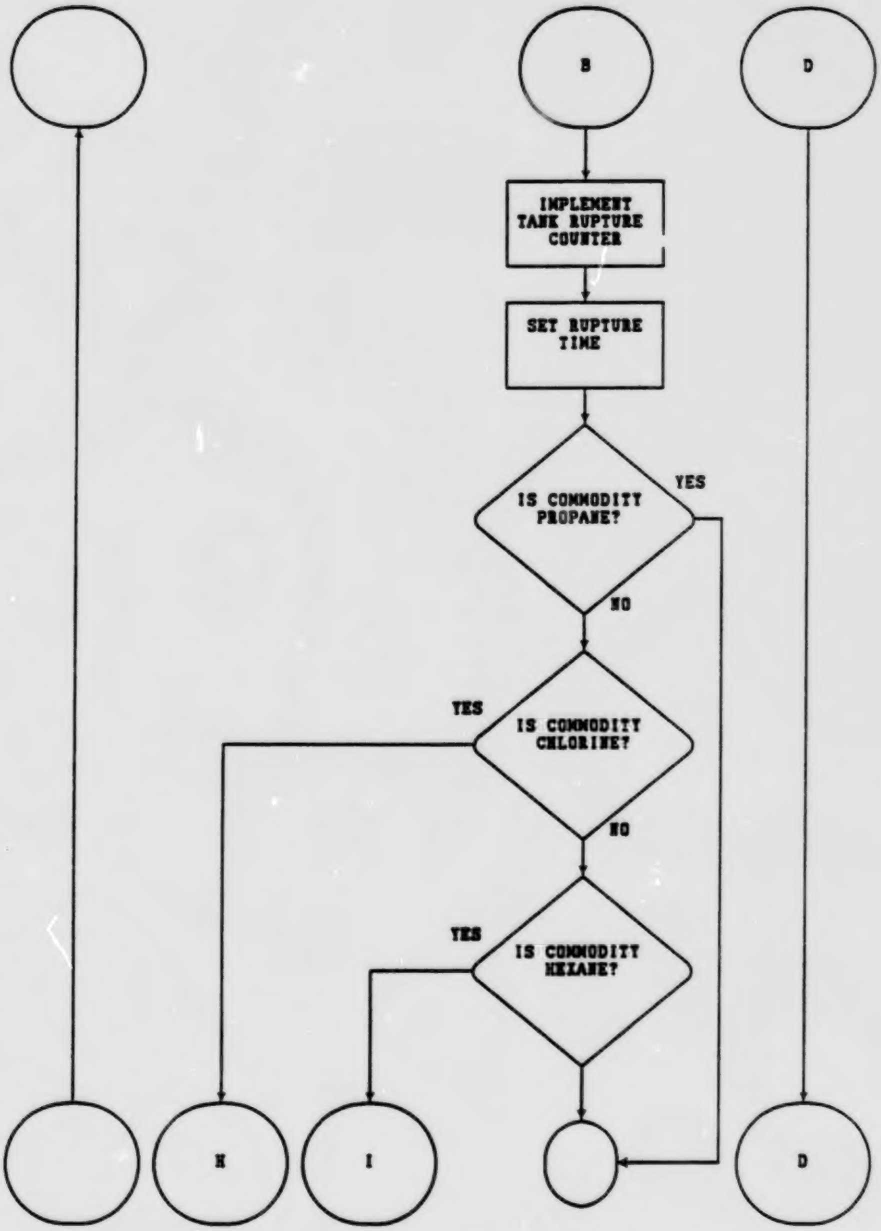

B
D
IMPLEMENT TANK RUPTURE COUNTER
SET RUPTURE TIME
IS COMMODITY PROPANE?
YES
NO
YES
IS COMMODITY CHLORINE?
NO
YES
IS COMMODITY HEXANE?
H
I
D

Figure 5.1 (cont'd) Flow Diagram for Simulation Software

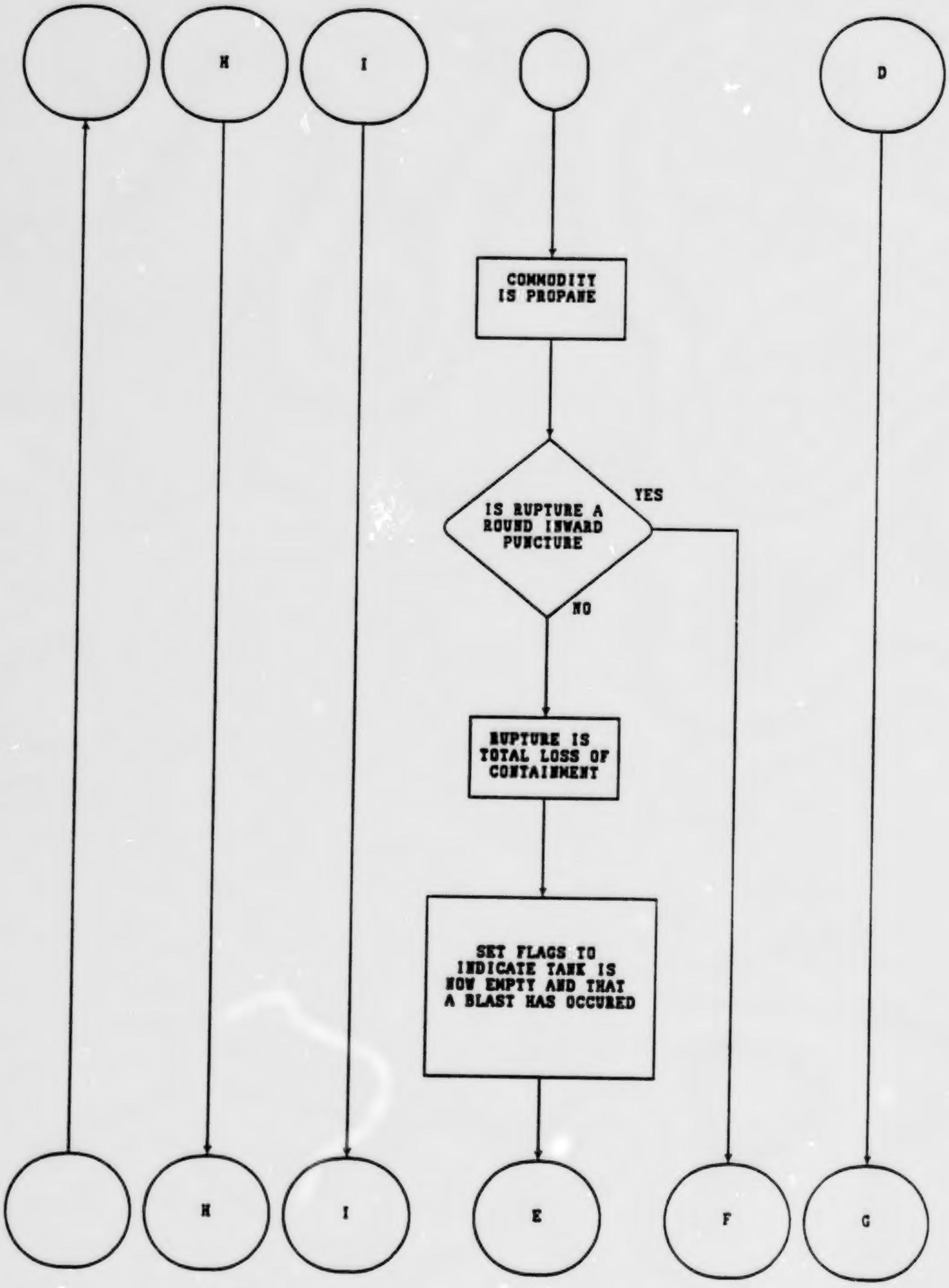

H
I
D
COMMODITY IS PROPANE
IS RUPTURE A ROUND INWARD PUNCTURE
YES
NO
RUPTURE IS TOTAL LOSS OF CONTAINMENT
SET FLAGS TO INDICATE TANK IS NOW EMPTY AND THAT A BLAST HAS OCCURED
H
I
E
F
G

Figure 5.1 (cont'd) Flow Diagram for Simulation Software

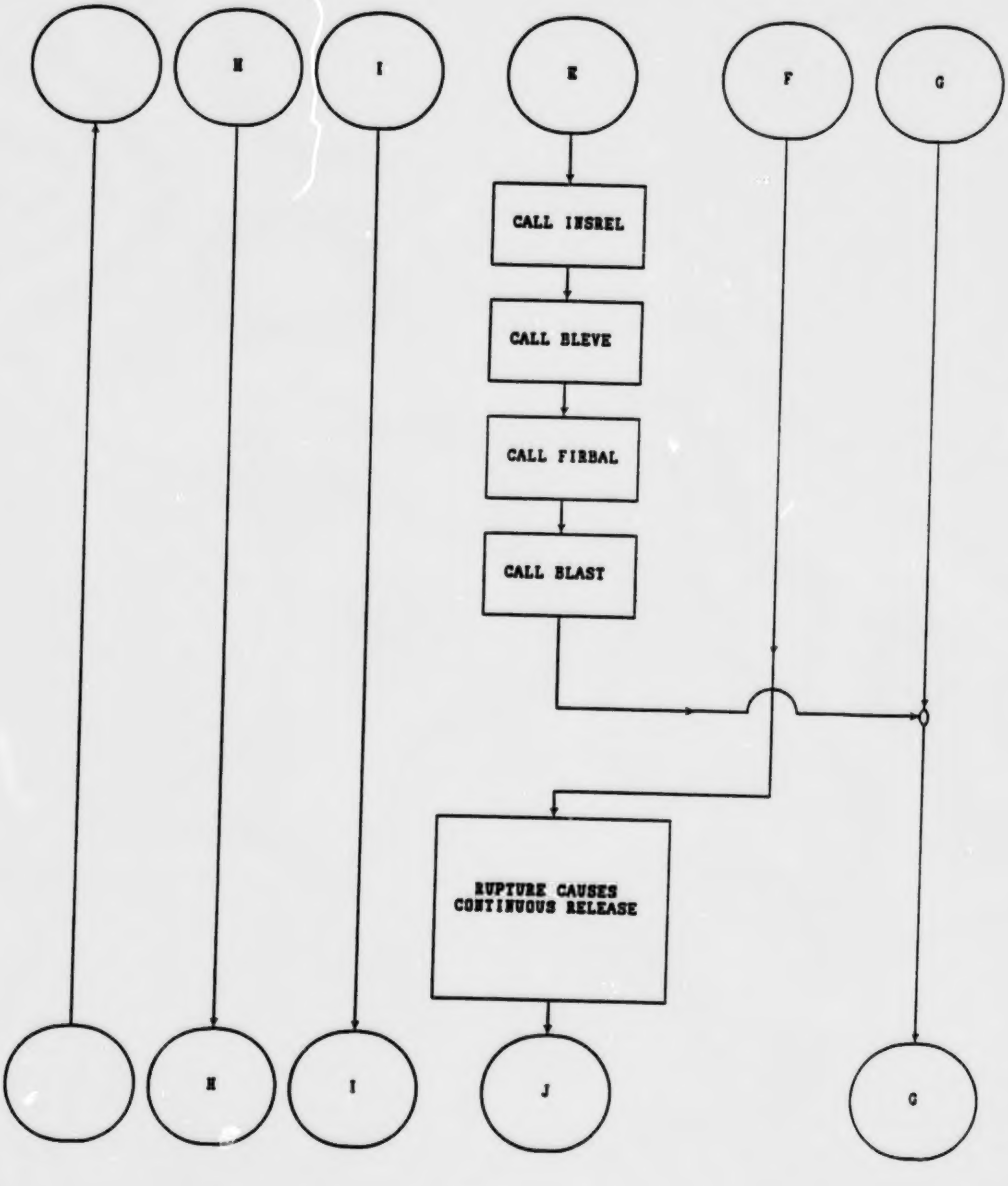

H
I
E
F
G
CALL INSREL
CALL BLEVE
CALL FIRBAL
CALL BLAST
RUPTURE CAUSES
CONTINUOUS RELEASE
H
I
J
G

Figure 5.1 (cont'd) Flow Diagram for Simulation Software

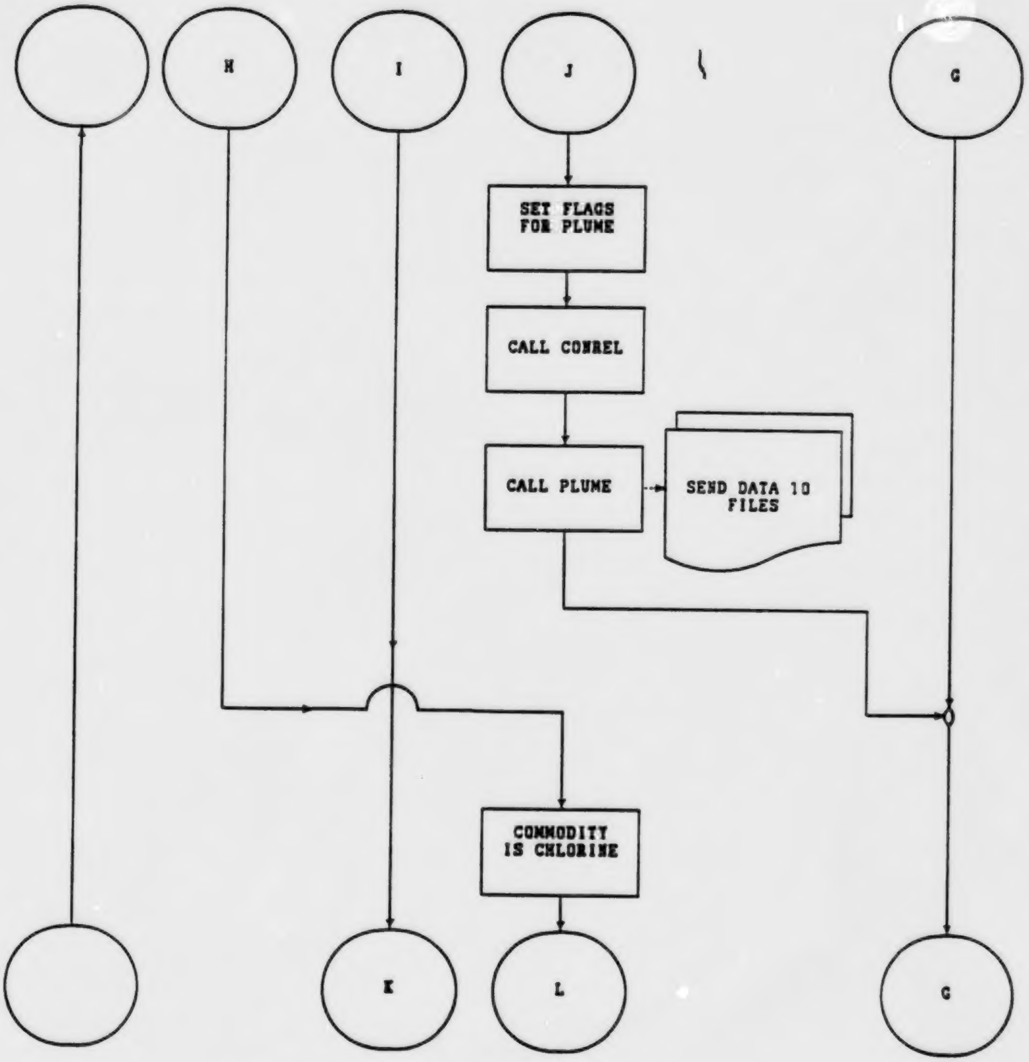

H
I
J
G
SET FLAGS FOR PLUME
CALL CONREL
CALL PLUME
SEND DATA TO FILES
COMMODITY IS CHLORINE
K
L
G

Figure 5.1 (cont'd) Flow Diagram for Simulation Software

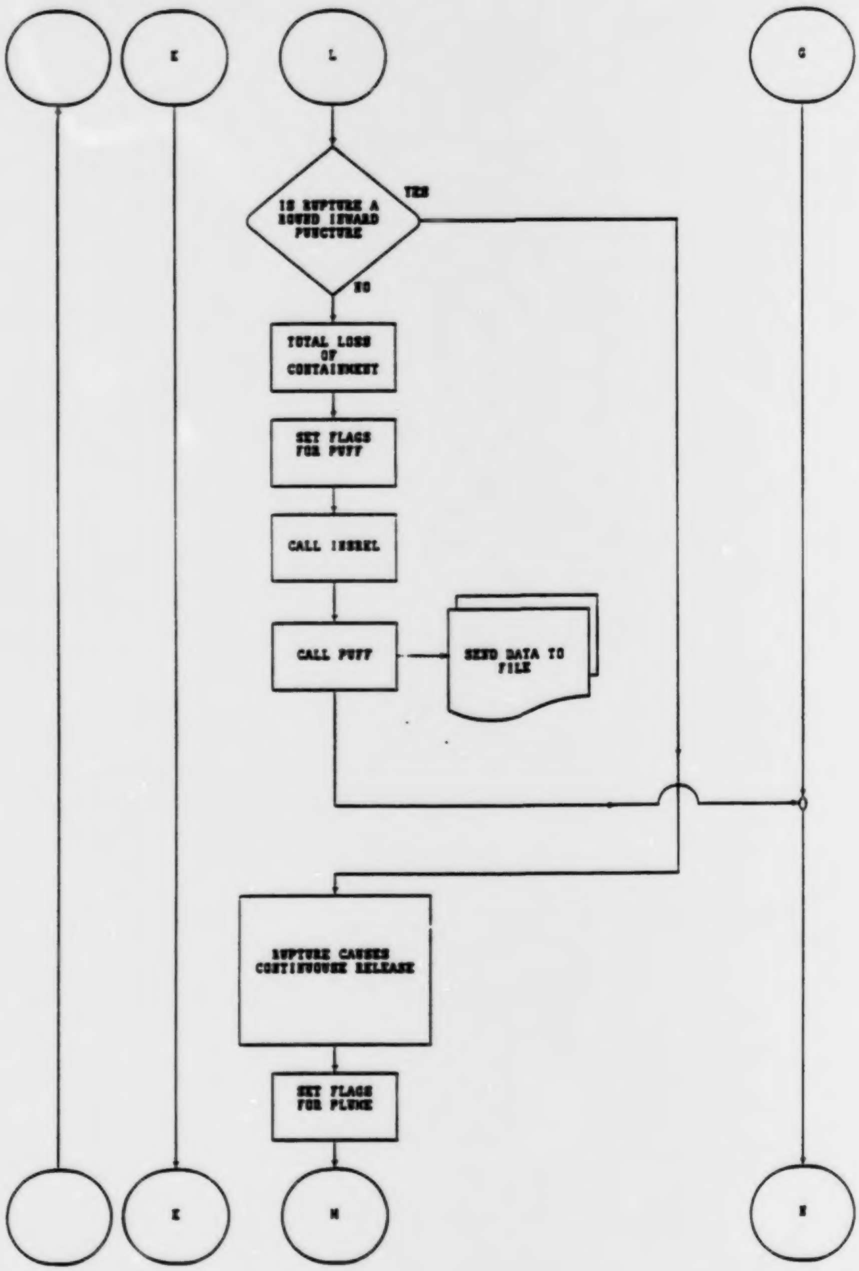

K
L
G
IS RUPTURE A ROUND INWARD PUNCTURE
YES
NO
TOTAL LOSS OF CONTAINMENT
SET FLAGS FOR PUFF
CALL INSREL
CALL PUFF
SEND DATA TO FILE
RUPTURE CAUSES CONTINUOUSE RELEASE
SET FLAGS FOR PLUME
K
M
E

Figure 5.1 (cont'd) Flow Diagram for Simulation Software

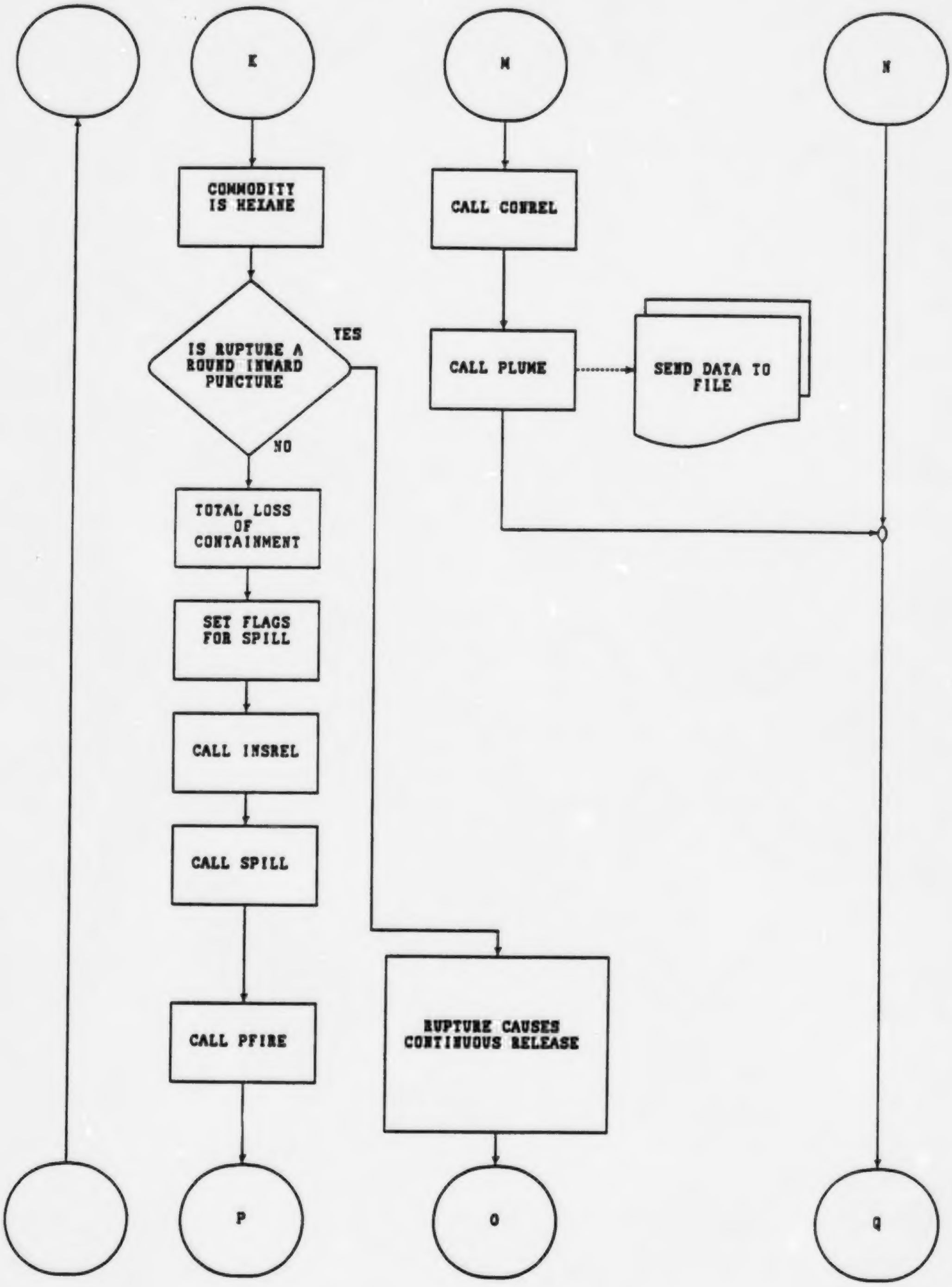

K
M
N
COMMODITY IS HEXANE
IS RUPTURE A ROUND INWARD PUNCTURE
YES
NO
TOTAL LOSS OF CONTAINMENT
SET FLAGS FOR SPILL
CALL INSREL
CALL SPILL
CALL PFIRE
CALL CONREL
CALL PLUME
SEND DATA TO FILE
RUPTURE CAUSES CONTINUOUS RELEASE
P
O
Q

Figure 5.1 (cont'd) Flow Diagram for Simulation Software

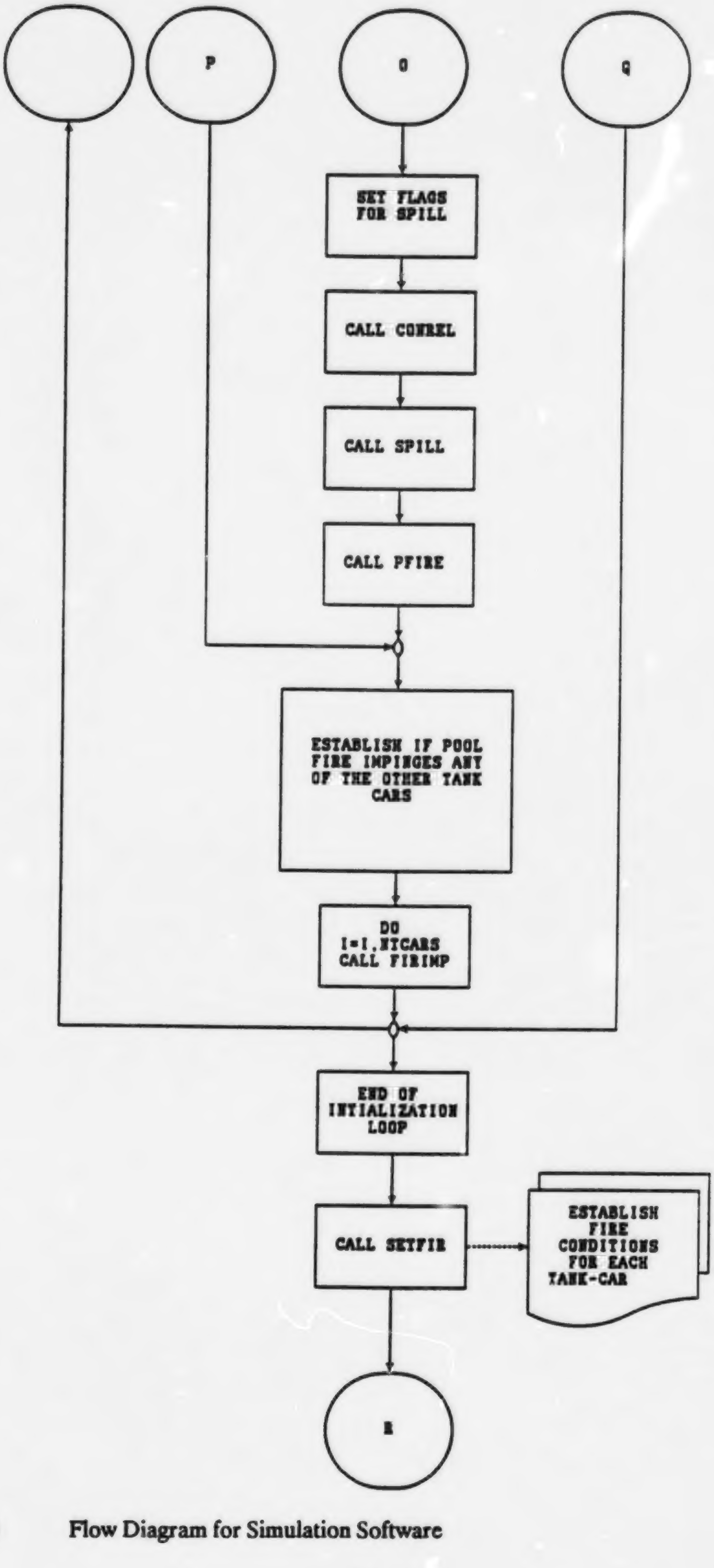

P
O
Q
SET FLAGS FOR SPILL
CALL CONREL
CALL SPILL
CALL PFIRE
ESTABLISH IF POOL FIRE IMPINGES ANY OF THE OTHER TANK CARS
DO I=1,NTCARS CALL FIRIMP
END OF INTIALIZATION LOOP
CALL SETFIR
ESTABLISH FIRE CONDITIONS FOR EACH TANK-CAR
R

Figure 5.1 (cont'd) Flow Diagram for Simulation Software

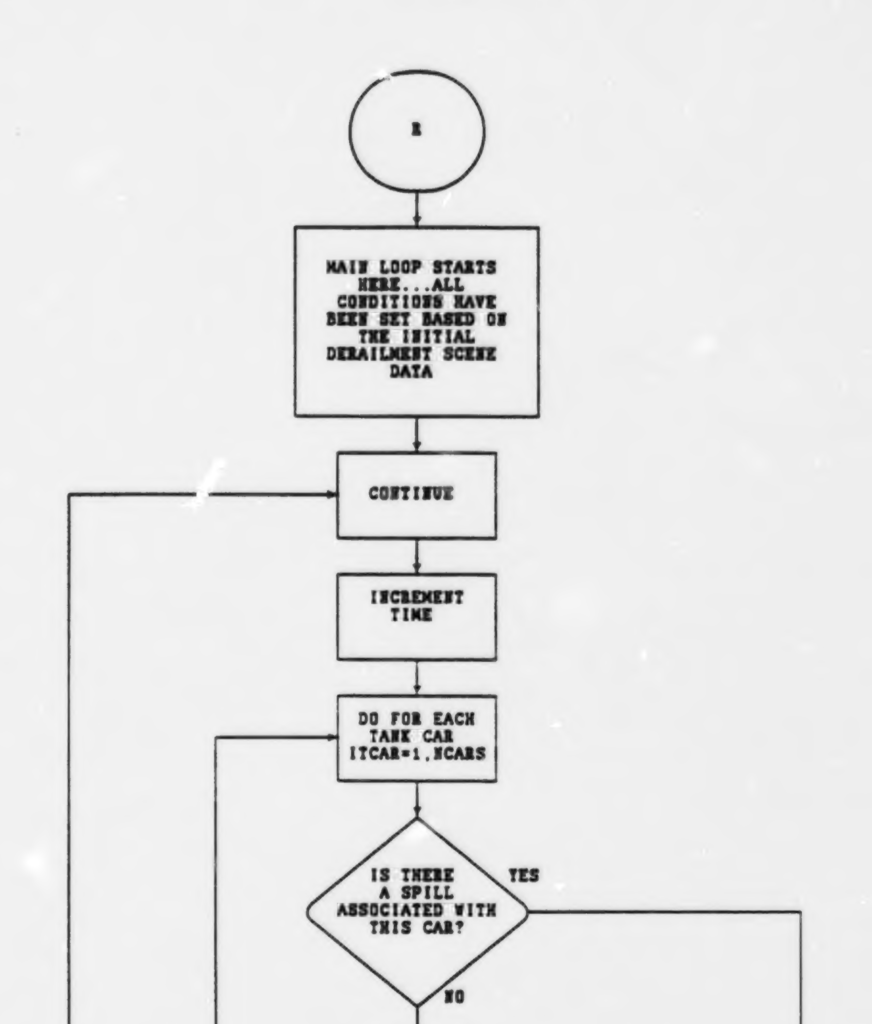

R
MAIN LOOP STARTS HERE...ALL CONDITIONS HAVE BEEN SET BASED ON THE INITIAL DERAILMENT SCENE DATA
CONTINUE
INCREMENT TIME
DO FOR EACH TANK CAR ITCAR=1,NCARS
IS THERE A SPILL ASSOCIATED WITH THIS CAR?
YES
NO
IS THIS CAR EMPTY?
YES
NO
IS THIS CAR RUPTURED?
YES
NO
X
T
U
V

Figure 5.1 (cont'd) Flow Diagram for Simulation Software

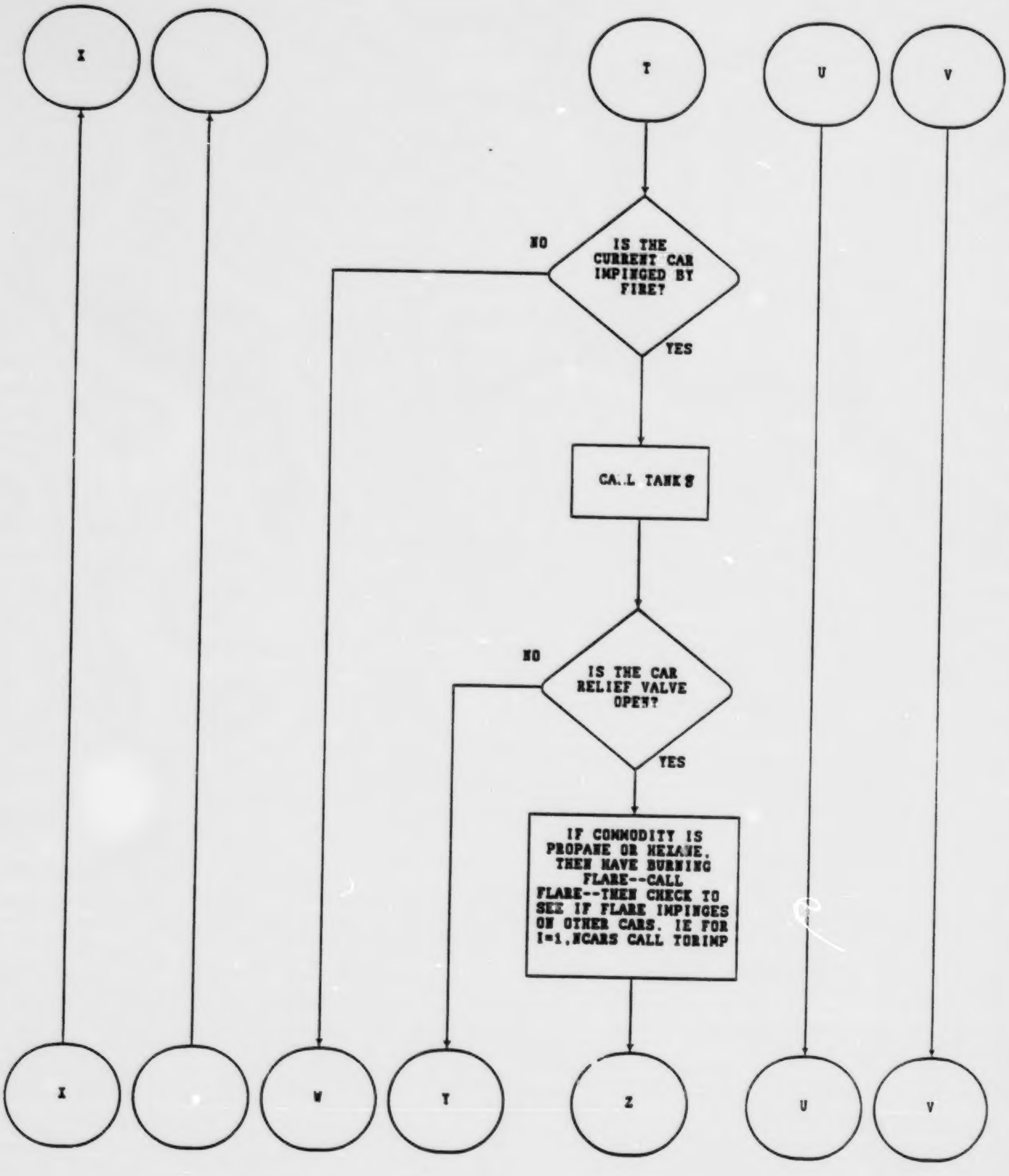

X
T
U
V
IS THE CURRENT CAR IMPINGED BY FIRE?
NO
YES
CALL TANKS
IS THE CAR RELIEF VALVE OPEN?
NO
YES
IF COMMODITY IS PROPANE OR HEXANE, THEN HAVE BURNING FLARE--CALL FLARE--THEN CHECK TO SEE IF FLARE IMPINGES ON OTHER CARS. IE FOR I=1,NCARS CALL TORIMP
X
W
T
Z
U
V

Figure 5.1 (cont'd) Flow Diagram for Simulation Software

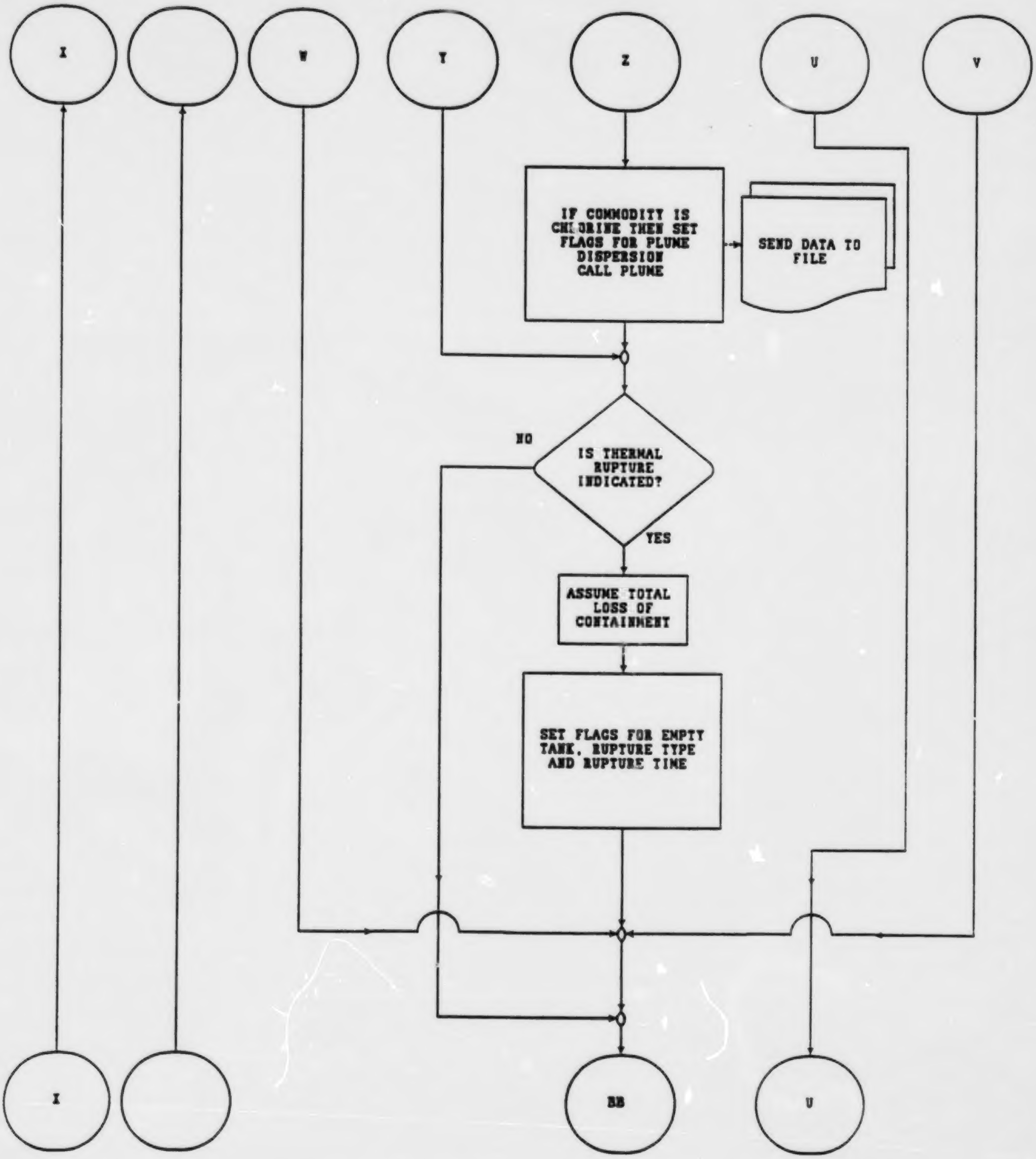

X
W
Y
Z
U
V
IF COMMODITY IS CHLORINE THEN SET FLAGS FOR PLUME DISPERSION
CALL PLUME
SEND DATA TO FILE
NO
IS THERMAL RUPTURE INDICATED?
YES
ASSUME TOTAL LOSS OF CONTAINMENT
SET FLAGS FOR EMPTY TANK, RUPTURE TYPE AND RUPTURE TIME
X
BB
U

Figure 5.1 (cont'd) Flow Diagram for Simulation Software

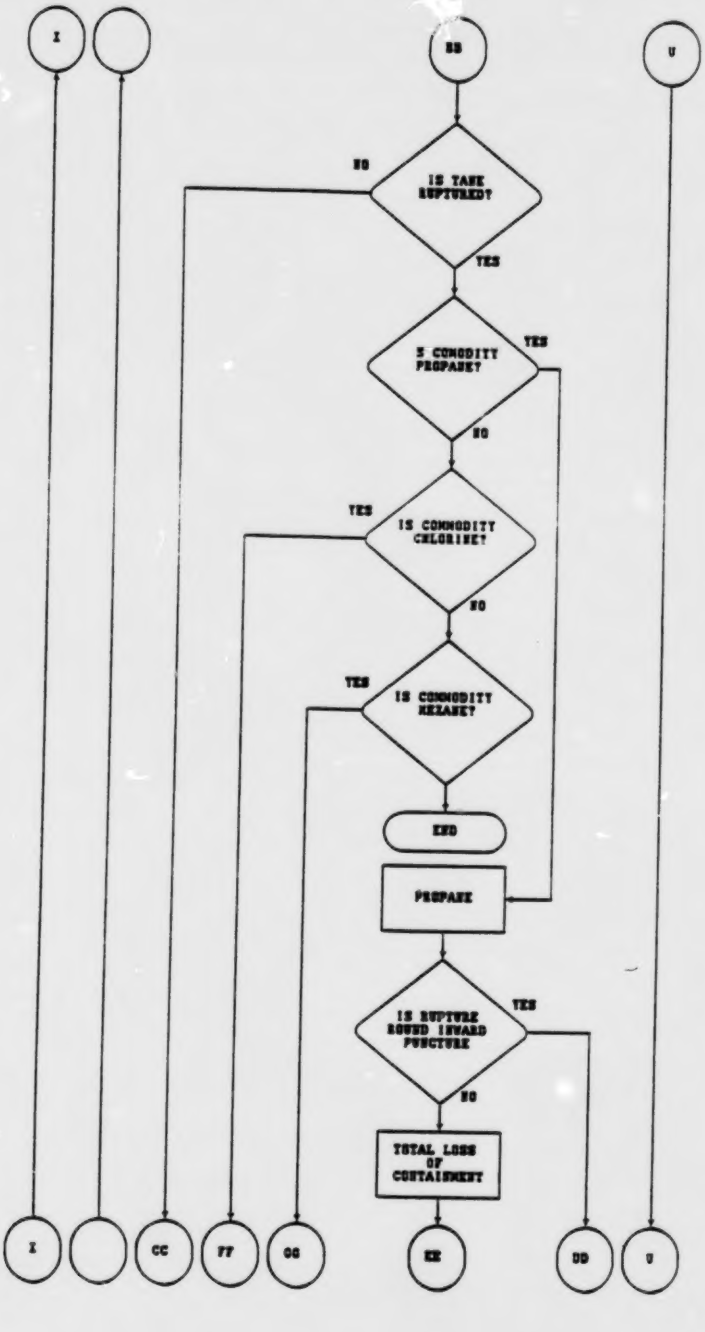

I
BB
U
NO
IS TANK RUPTURED?
YES
S COMODITY PROPANE?
YES
NO
YES
IS COMMODITY CHLORINE?
NO
YES
IS COMMODITY HEXANE?
END
PROPANE
IS RUPTURE ROUND INWARD PUNCTURE
YES
NO
TOTAL LOSS OF CONTAINMENT
I
CC
FF
GG
EE
DD
U

Figure 5.1 (cont'd) Flow Diagram for Simulation Software

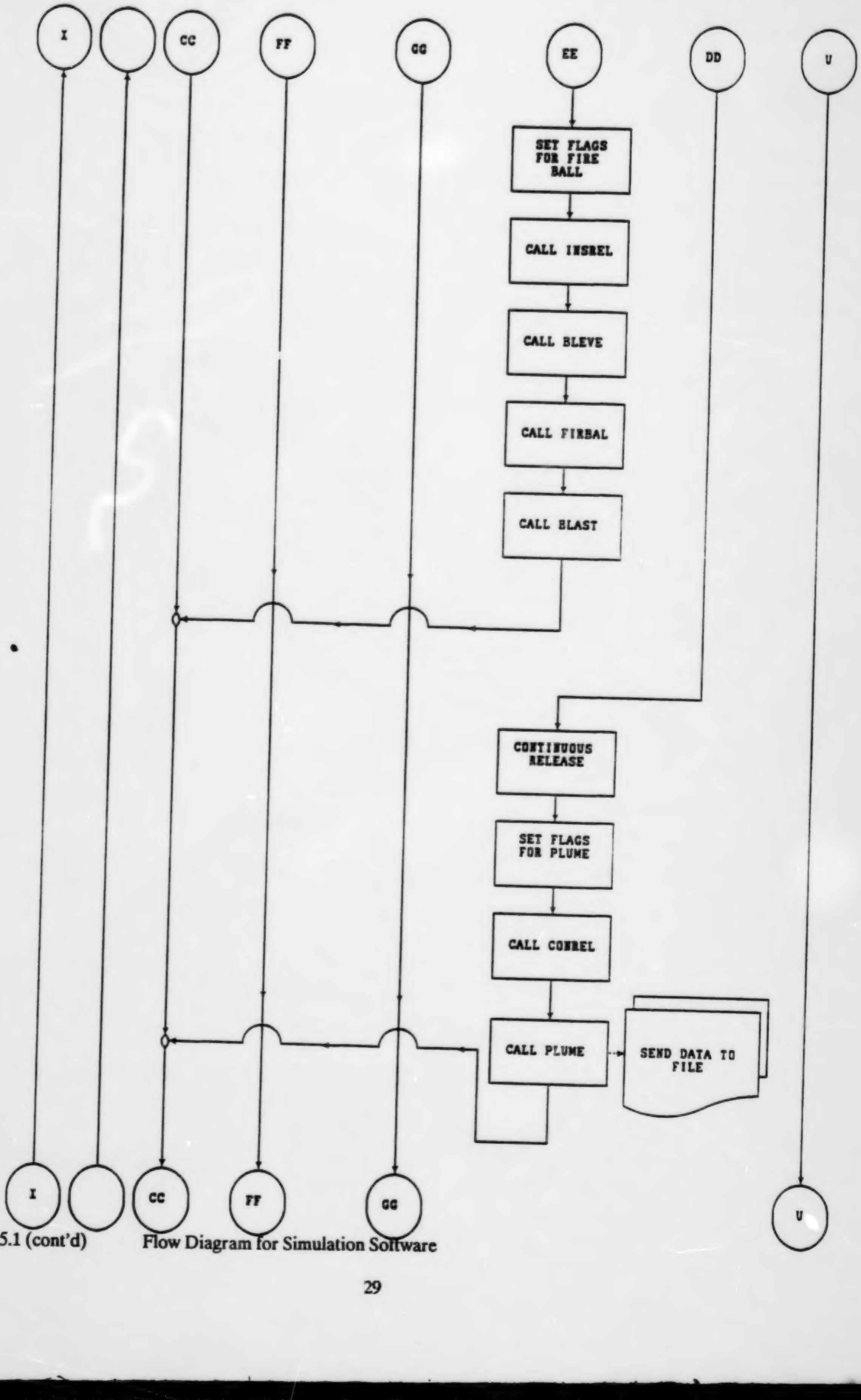

I
CC
FF
GG
EE
DD
U
SET FLAGS FOR FIRE BALL
CALL INSREL
CALL BLEVE
CALL FIRBAL
CALL BLAST
CONTINUOUS RELEASE
SET FLAGS FOR PLUME
CALL CONREL
CALL PLUME
SEND DATA TO FILE
I
CC
FF
GG
U

Figure 5.1 (cont'd) Flow Diagram for Simulation Software

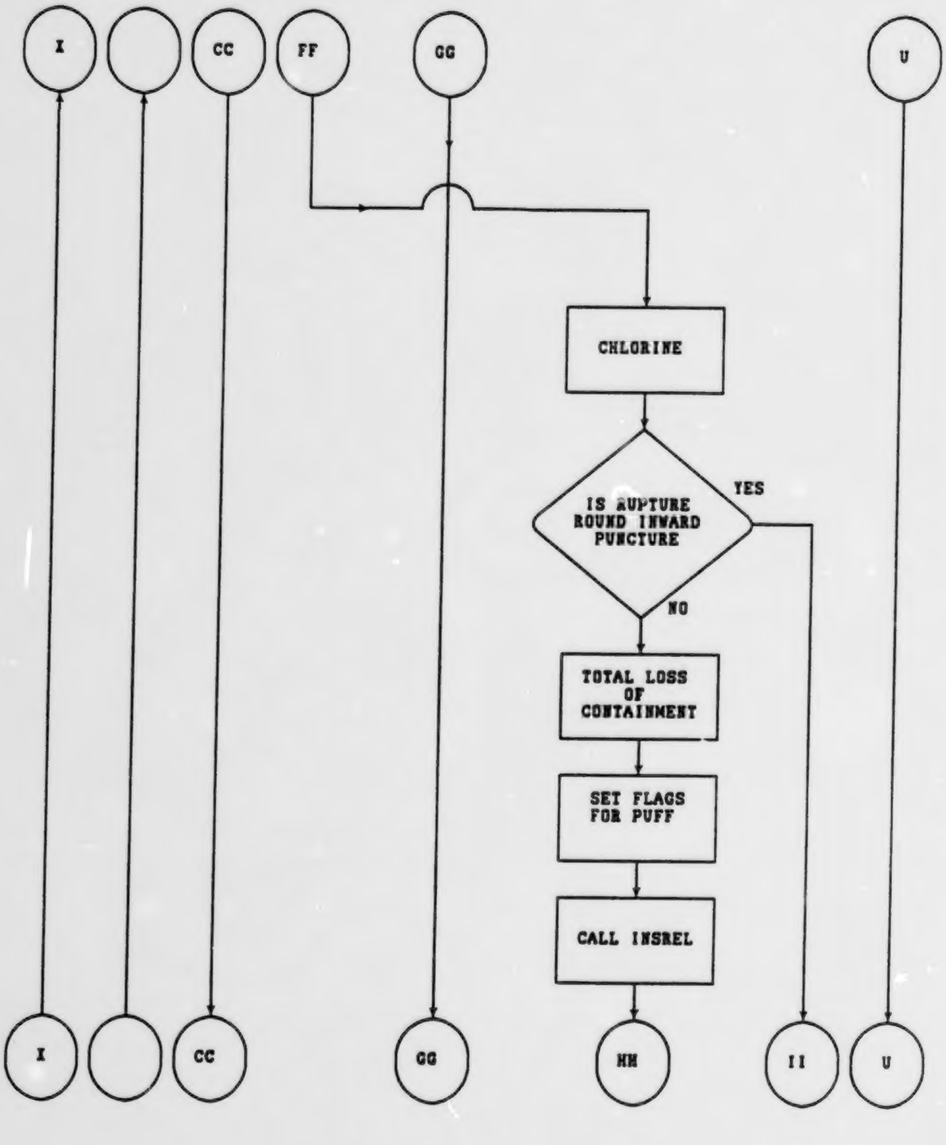

I
CC
FF
GG
U
CHLORINE
IS RUPTURE ROUND INWARD PUNCTURE
YES
NO
TOTAL LOSS OF CONTAINMENT
SET FLAGS FOR PUFF
CALL INSREL
I
CC
GG
HH
II
U

Figure 5.1 (cont'd) Flow Diagram for Simulation Software

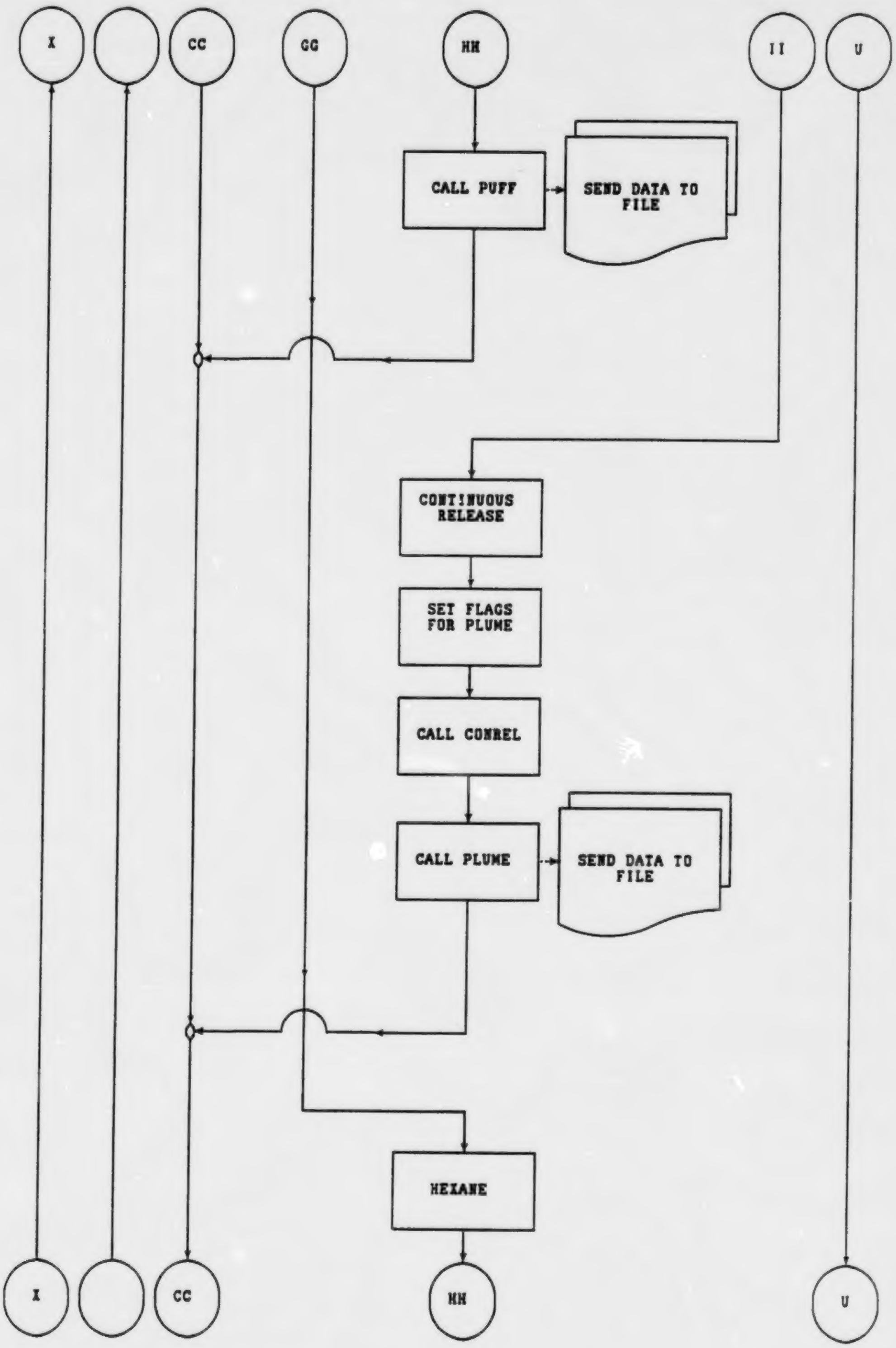

X
CC
GG
HH
II
U
CALL PUFF
SEND DATA TO FILE
CONTINUOUS RELEASE
SET FLAGS FOR PLUME
CALL CONREL
CALL PLUME
SEND DATA TO FILE
HEXANE
X
CC
HH
U

Figure 5.1 (cont'd) Flow Diagram for Simulation Software

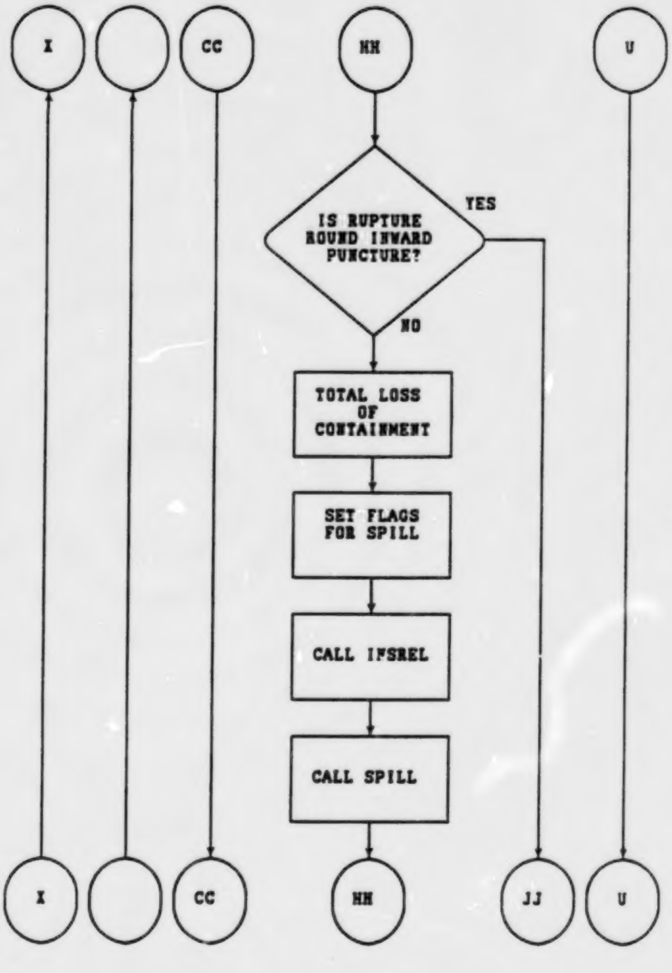

X
CC
HH
U
IS RUPTURE
ROUND INWARD
PUNCTURE?
YES
NO
TOTAL LOSS
OF
CONTAINMENT
SET FLAGS
FOR SPILL
CALL IFSREL
CALL SPILL
X
CC
HH
JJ
U

Figure 5.1 (cont'd) Flow Diagram for Simulation Software

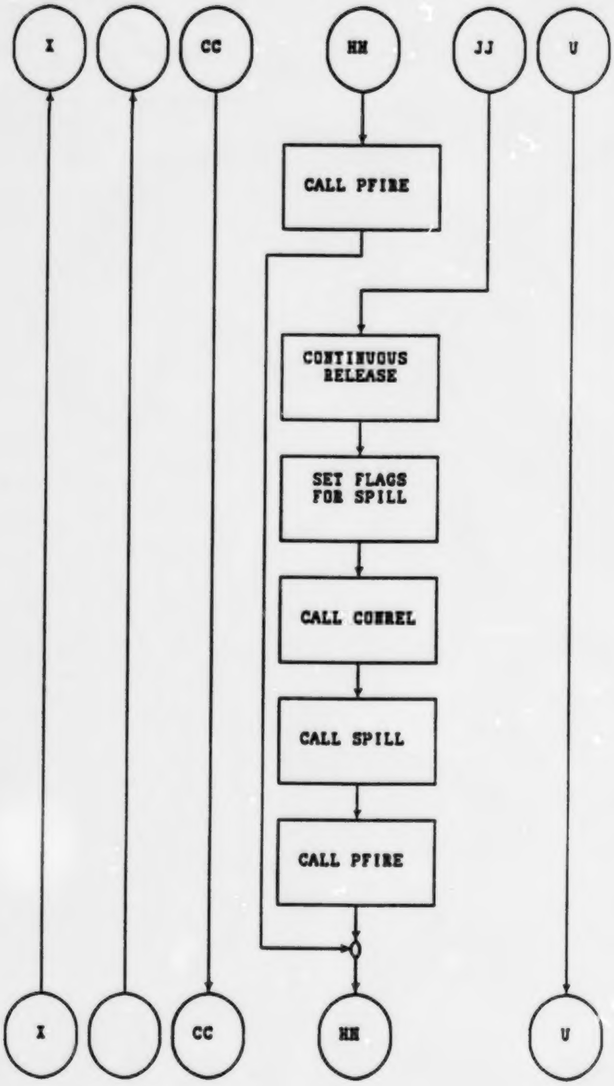

I
CC
HH
JJ
U
CALL PFIRE
CONTINUOUS RELEASE
SET FLAGS FOR SPILL
CALL CONREL
CALL SPILL
CALL PFIRE
I
CC
HH
U

Figure 5.1 (cont'd) Flow Diagram for Simulation Software

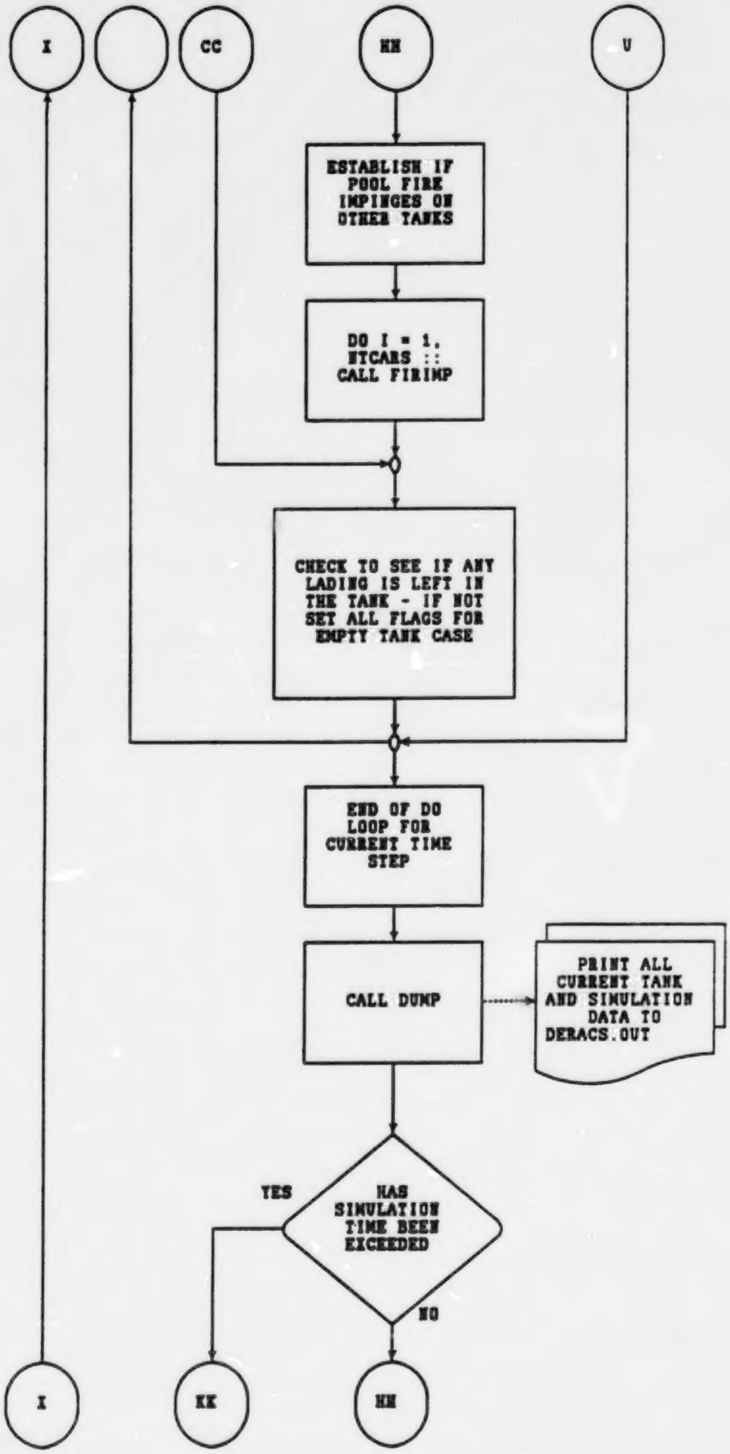

I
CC
HH
U
ESTABLISH IF POOL FIRE IMPINGES ON OTHER TANKS
DO I = 1, NTCARS :: CALL FIRIMP
CHECK TO SEE IF ANY LADING IS LEFT IN THE TANK - IF NOT SET ALL FLAGS FOR EMPTY TANK CASE
END OF DO LOOP FOR CURRENT TIME STEP
CALL DUMP
PRINT ALL CURRENT TANK AND SIMULATION DATA TO DERACS.OUT
YES
HAS SIMULATION TIME BEEN EXCEEDED
NO
I
KK
HH

Figure 5.1 (cont'd) Flow Diagram for Simulation Software

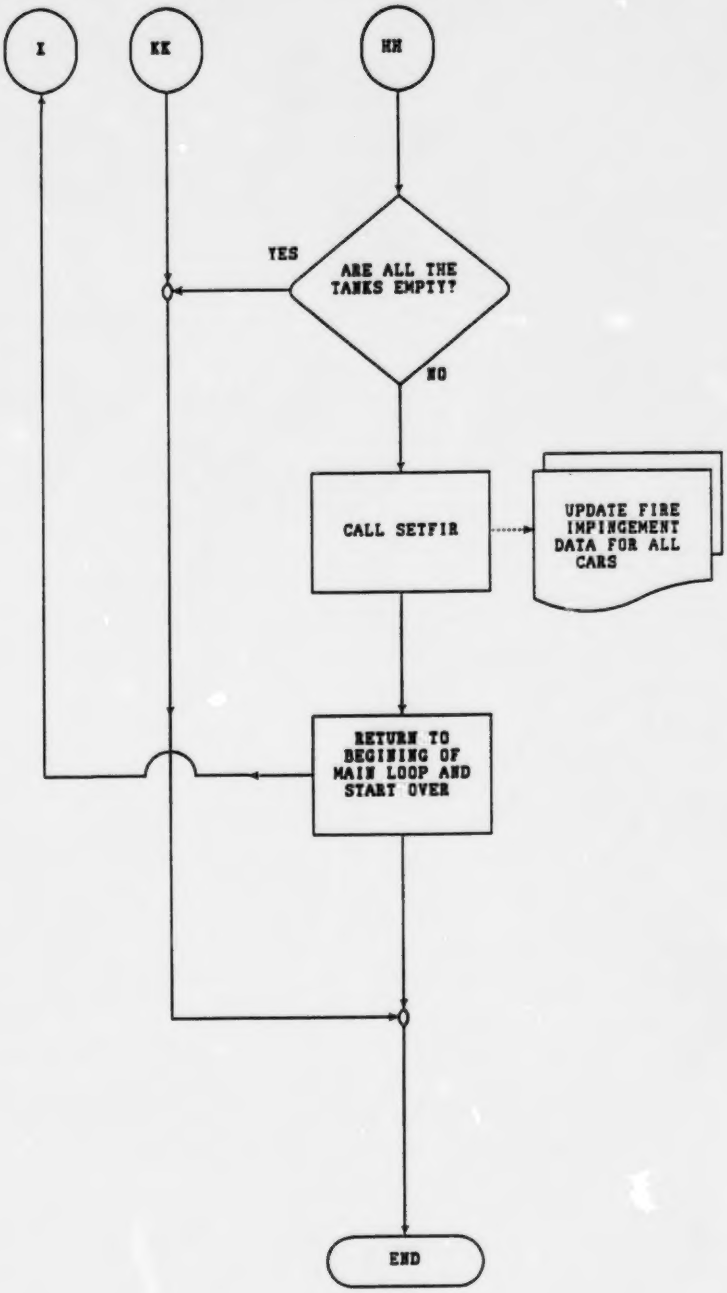

I
KK
HH
YES
ARE ALL THE TANKS EMPTY?
NO
CALL SETFIR
UPDATE FIRE IMPINGEMENT DATA FOR ALL CARS
RETURN TO BEGINING OF MAIN LOOP AND START OVER
END

Figure 5.1 (cont'd)   Flow Diagram for Simulation Software

The third and final commodity type considered in DERACS is hexane. If the current tank is established as having contained hexane then again the rupture type is established. If it is determined that there is a total loss of containment then it is assumed that a pool of the spilled hexane will collect and that a pool fire will result. If it is established that there is a continuous release due to round inward puncture then again it is assumed that a pool will form and that a pool fire will result. Subroutines are called to establish the volume of the released hexane and the size of the resulting pool. The fire effects of the pool fire are calculated for display by the post-processor. Since a pool fire has been established the next series of lines of code are used to determine if this pool fire is impinging on any of the other tanks in the simulation. A do-loop considers each of the other tank cars in turn and in this do-loop the subroutine FIRIMP is called. This subroutine determines if the tank is effected by the pool fire and to what degree, whether it is a fully engulfing situation or a partially engulfing situation. If fire impingement is indicated then the various flags are set indicating that the tank is effected by the fire and this information is stored for later analysis to determine the final fire mode of each car.

This is now the end of the initialization block in the DERACS main program. At this point it is now established which of the tanks have ruptured which of these ruptures has resulted in explosions or toxic releases and which of these ruptures have resulted in pool fires which impinge on other tanks. The final line of the block is a call to subroutines SETFIR. This subroutine is used to analyze the fire impingement matrix for the tanks. The algorithms in SETFIR use a priority system to establish the final fire type for each car that is impinged by fire and the critical data that is to be used in the tank car thermal model.

This marks the end of the main initialization loop. Up until now only derailment data has been used to set conditions of the various tanks. From this point on in the program the simulation steps through time to predict the consequences of the established fires, plumes, puffs etc.

The next block of the DERACS program is the main iteration loop. The main loop of the DERACS program is where the time integration of the derailment accident takes place. At the beginning of this main loop the global time is incremented by the time step. Then the main do-loop in this block takes over. This is a do-loop similar to the one in the initialization loop that steps through each tank car in the simulation. For each tank car it is determined whether the tank is empty or whether it has ruptured and if it has ruptured what consequences will take place.

If the tank has not ruptured and is impinged by fire then the tank car thermal model TANK8 is called. The tank car thermal model is of course the submodel which accounts for the effects of fire impingement, either engulfing, partially engulfing, or torch type, on the response of the tank and its lading. After returning from the tank car thermal model it is determined whether or not the current tank has its relief valve in operation. If the current tank contains propane or hexane then it is assumed that a burning relief valve flare will be present. It is then established whether or not this burning relief valve flare impinges on any other tanks. If however, the current tank contains chlorine then the relief valve action will result in the discharge of toxic gases into the environment and therefore the plume tracking subroutine will be called up.

The next block of code is used to determine whether or not stresses in the tank wall of the fire impinged tank are such that thermal rupture is likely. If thermal rupture is indicated then it is assumed that there will be a total loss of containment. This will result in the total release of the remaining lading in the tank. If the commodity of the tank that has just ruptured is propane then it is assumed a fire ball will result. The calculations will proceed to check to see if a BLEVE is likely or not and the consequences of the blast will be determined including the effects of blast waves and thermal radiation from the fire ball. If the commodity is chlorine then it is assumed a puff release will occur and the puff tracking subroutines will be called into action. If the commodity is hexane then it is assumed that the total loss of containment will result in the formation of a flammable pool and a pool fire will result. Calculations are performed to establish whether or not this pool fire impinges on any other tanks in the simulation.

All of the above calculations take place if the tank is impinged by fire and the tank car thermal model is used. However, if the tank has already ruptured then the tank car thermal model is not called and this block of code is bypassed. In the event that the tank has ruptured it is determined whether the rupture was a total loss of containment type rupture or whether it was a continuous release type. Based on the commodity and the rupture type the consequences are determined as previously explained in the initialization loop. This brings the main do-loop in the main iteration loop to an end.

The next statement following the end of the main do-loop is a call to a subroutine DUMP which is used to print all of the current tank, fire, explosion, plume etc. data to file. The file name is DERACS.OUT. This of course is stored for later use by the DERACS post-processor.

The next statements involve checks to see if the simulation time has been exceeded or if all of the tank cars are empty. If either of these are true then the program terminates. If the program continues subroutine SETFIR is called to update the fire impingement conditions for all the tanks and then a go to statement sends control back to the beginning of the main loop where time is incremented and the main do-loop is again set into action.

5.3 Primary Subroutines

The simulation software is made up from a number of subroutines which contain the various submodels that represent the processes that are taking place during the accident. The primary subroutines and their basic function are summarized in Table 5.1. These subroutines then call a number of lower level routines.

Table 5.1 Summary of Primary Simulation Subroutines Called by DERACS

FUNCTION	SUBROUTINE NAMES
I/O	DUMP, SIMCON
Derailment Mechanics	DERAIL, IMPACT
Liquid and Gaseous Releases	SPILL, INSREL, CONREL
Fire Impingement	PFIRE, FIRIMP, FLARE, TORIMP, SETFIR
Tank Thermal Model	TANK8
Explosions	FIRBAL, BLEVE, BLAST
Plume and Puff Dispersion	PLUME, PUFF

The primary subroutines will now be described separately. The source code for the these subroutines can be found in Appendix B.

a. Subroutine SIMCON

This subroutine is used to read in the initial conditions for the accident. This basic information is needed to perform the simulation. The name SIMCON stands for SIMulation CONditions.

SIMCON reads in information from two files named CARDATA.DAT and DERAIL.DAT. CARDATA.DAT is created by the DERACS preprocessor and DERAIL.DAT is generated either by the derailment and impact rupture routines or by the preprocessor depending on the option selected by the user.

CARDATA.DAT contains the environmental conditions such as wind speed and direction, air temperature and atmospheric pressure as well as train data for up to forty cars. The train data contains train speed and the track radius of curvature at the point of derailment and the car, truck, axle number of the car that initiates the derailment. This data is then followed by the total number of cars, and then for each car the car number and type. If the car is a tank car then additional information is read including the commodity type, initial fill level, initial temperature and whether the tank is insulated or not. SIMCON sorts through the cars in order and identifies up to eight tank cars by number (DERACS is presently limited to eight tank cars but can be expanded up to memory limits). The associated tank car data is then stored by tank car number.

DERAIL.DAT contains the post derailment data for the cars. This includes the car position and orientation, and the tank rupture data. The tank position and orientation is given relative to a local coordinate system centered at the point of derailment. This coordinate system is oriented such that the x axis points in the direction of the train motion and is tangent to the track at the point of derailment, and the y axis is perpendicular to the track pointing to the outside of the curve. For straight track the y axis defaults to satisfy the right hand rule where the z axis points up out of the page. The rupture data contains the type of rupture (no rupture, total loss of containment or round inward puncture) and if appropriate the rupture diameter and position along the length of the tank.

With this information collected by SIMCON it is then possible to carry out the time integration of the accident.

b. Subroutine DUMP

This subroutine dumps information to a file called DERACS.OUT after each simulation time step has been completed. This information pertains to the tank-cars involved in the derailment accident and includes all the information needed by the post-processor to create a graphic image or text summary of the derailment. The information is printed out by tank-car and contains information as summarized in Table 5.2. The variables are defined in the source code in Appendix B.

Note that DUMP does not print out all of the plume and puff information. The plume and puff data is handled separately because of the large volume.

c. Subroutine SPILL

This subroutine is used to calculate the approximate diameter of a spill of flammable liquid for the purposes of estimating the duration and consequences of pool fires. In DERACS it is assumed that if the commodity hexane is spilled due to an impact induced rupture then a pool will form and will ignite to render a pool fire. This pool fire then may impinge on other tanks with the associated consequences.

This subroutine is based on the assumption that the ground at the spill location is slightly concave with a specified radius of curvature. When the liquid is spilled it will then form a pool of a certain diameter and depth based on volume considerations. This approach was used because it was considered to be more appropriate for the DERACS simulation than some of the simple spill models available in the open literature. Some of these other spill models tend to assume smooth flat surfaces which then give pool diameter estimates which are very large and burn times that are short. With the present model burn times can be calibrated by choosing an appropriate ground curvature.

Table 5.2 Summary of Data Printed to File By Dump

Line	Variables
1	icar, itype, xcar, ycar, angcar, rolcar
tank car data	
2	itcar, icomm, ifprot, temp, fill
3	ptank, mass, tfs, tfc, theta
4	sigin, sigout, sult
5	qfc, qfs, qg, qdotg, qdotf
6	mdot, t0, t90, t180
fire data	
7	ifire, fdata, timin, timout, timnow
rupture data	
8	ruptyp, tirupt, ruptd, ruptl
spill data	
9	ispill, tispl, dspill, dpt1, dpt2, dpt3
flare data	
10	iflare, tiflar, toflar, flarel
blast data	
11	iblast, tiblast, db1, db2, db3, dbt1, dbt2, dbt3
plume data	
12	iplume, tiplm, toplm
puff data	
13	ipuff, tipuff

During the simulation SPILL is called every time step if it is indicated that a spill is associated with a tank car. The current size of the spill is calculated based on current spill volume considerations. After each time step the spill volume is reduced by using an assumed constant surface regression rate.In this way the spill diameter diminishes with time. As the pool diameter decreases the pool fire thermal effects on the surroundings and other tanks are reduced.

Technical data for subroutine SPILL can be found in Appendix D.9.

d. Subroutine CONREL

This subroutine is used to calculate the mass release rate of vapor and liquid for a continuous type release (CONtinuous RELease). This subroutine and its associated lower level routines were taken from the program LEAKR of Reference [1].

The program LEAKR is capable of predicting the continuous release rates for a wide range of commodities and release scenarios. In its original form LEAKR performs a time integration of the release sequence to give a summary of mass flow rates during different phases of the release. This includes liquid and two-phase flow rates when the liquid is above the rupture opening, and vapor flow rates when the liquid is below the rupture opening. The program accounts for heat transfer from the surroundings and the effects of evaporation on the temperature of the remaining liquid in the tank. By accounting for the various phases of the continuous release LEAKR gives an estimate of the time to empty the tank of its contents.

The program LEAKR was significantly modified for the present purposes. The modified version of LEAKR is called CONREL. CONREL is given the basic information about the rupture including the rupture size and location on the tank. In addition to the rupture data CONREL requires tank and surroundings data such as wind speed, air temperature, the tank lading type. the liquid level and mean temperature, and the tank pressure. This information is then used to calculate the current mass flow rate due to either liquid, 2- phase or vapor flow. These flow rates are stored and the mass is reduced by the product of the mass flow rate and the time step. THe stored mass flow rates are used to calculate liquid pool sizes and plume dispersion.

Further technical details of CONREL can be found in Appendix D.4.

e. Subroutine INSREL

Subroutine INSREL is used to calculate the mass released in an instantaneous release. An instantaneous release is defined here as one that take place in less than one time step. The amount of mass released depends on the rupture mode.

If the rupture is a total loss of containment, then it is assumed that all the mass in the tank at rupture is released. This in fact is not what happens in reality. In an actual loss of containment situation a portion of the lading suddenly expands from a liquid to a vapor when the tank is opened. Depending on the situation the portion expanded can be as high as 60% as discussed in Reference [2]. The remainder of the liquid then boils off over a period of time. These two phases of a total loss of containment have been combined into a single event for the present simulation. This is a significant simplification and will be addressed later when discussing model weaknesses.

If the lading is propane, the release is assumed to result in a BLEVE or fireball. If the lading is chlorine, then the release is modelled as a puff. If the lading is hexane, then the release forms a spill.

If the mode of rupture is a round inward puncture, then INSREL will be called if the lading is hexane. In this case, only the liquid above the hole is assumed to spill. The remaining liquid is assumed to be trapped in the tank.

For details of this calculation, the reader is directed to the source code listings in Appendix B.

f. Subroutine PFIRE

This subroutine is called to predict the thermal consequences of a pool fire. The diameter of the pool is determined in Subroutine SPILL each time step and then PFIRE is called to calculate the radiant heat transfer to remote targets. The emissive power of the pool fire is calculated based on an empirical relation for large pool fires involving hydrocarbon fuels. The heat transfer calculations are based on the assumption that the pool fire shape can be approximated by a cylinder with a diameter equal to the pool fire diameter and a height based on the height of large fires. With this assumption it is possible to calculate the radiant flux as a function of distance from the cylinder centre. For the purposes of the DERACS simulation three levels of flux considered significant have been established (they are 1st 2nd and 3rd degree burns to humans exposed for more than 1 minute). Subroutine PFIRE calculates the distances from the pool centre to the point where the above levels of exposure are expected. These distances are then stored in a file for later use by the post processor for displaying the consequences of the fires.

Technical details of the PFIRE calculations can be found in Appendix D.8.

g. Subroutine FIRIMP

This subroutine is used to determine if a given pool fire impinges on any of the other tanks in the simulation. When a tank has been identified as being ruptured and surrounded by a burning pool this subroutine is called for each of the other tanks in the simulation to determine whether they are engulfed or partially engulfed by the fire. If any of the other tanks are impinged by the pool fire then the variables IFIREM and FDATAM are set appropriately. IFIREM and FDATAM are matrices that store the effects of all fires on all of the tanks in the simulation. These matrices are later used by the subroutine SETFIR to determine the fire impingement cases for each tank in the simulation.

Details of the calculation procedure can be found in Appendix D.6.

h. Subroutine FLARE

Subroutine FLARE is used to determine the length characteristics of a burning relief valve flare for the purposes of establishing if any other tanks are impinged by torch fires. To determine if a tank is impinged by a flare emanating from another tank it is necessary to know the approximate length of the flare along the ground (ie. the horizontal length). In this subroutine all tanks that are rolled between 80 and 100 degrees from the vertical are considered as possible sources of impinging torches (all other flares are ignored). The flare length is based on experimental data for vertical jet flames and for horizontal jet flames where buoyancy has significant effects on the shape of the flames (ie. buoyancy will tend to make a horizontal flame jet curve upwards thereby reducing its horizontal length).

Technical details of calculations performed in FLARE can be found in Appendix D.7.

i. Subroutine TORIMP

Subroutine TORIMP (TORch IMPingement) is used to establish if any tanks are impinged by a flare (torch) from a given tank. It performs the same function as FIRIMP does for the case of pool fires. By knowing the flare length, position and direction for a given tank it is possible to determine whether it impinges on any of the other tanks in the simulation. If torch impingement takes place then the variables IFIREM and FDATAM are set appropriately. In this case the value of FDATAM represents the impingement distance from the flare tank to the impinged tank.

Technical details of calculations performed in TORIMP can be found in Appendix D.10.

j. Subroutine SETFIR

After each time step SETFIR is called to review the fire impingement matrices IFIREM and FDATAM to determine the fire impingement case for each tank in the simulation. Since each tank may be affected by more than one of the other tanks in the simulation a fire impingement priority system was established. The basic rules of this system are as follows:

i) a tank can only be impinged by a torch or a pool fire but not both (this assumption was necessary because the tank car thermal model cannot presently handle both types of fire impingement simultaneously).

ii) once a fire impingement type (torch or pool fire) has been established for a tank it cannot be changed during the simulation (again this is a present limitation of the tank car model).

iii) torch fire impingement takes priority over pool fire impingement (this was chosen because a torch fire can result in very rapid failure of a tank and therefore was considered more severe for the present version of DERACS).

iv) the final fire impingement effect is based on the single worst case fire impingement (ie. if the current tank fire case is torch impingement then if several torches are impinging the current tank the thermal calculations are based on the single torch which is closest). In other words the worst case and not the sum of all cases is considered for the thermal calculations. This approach is correct for the case of fully engulfing fires and for some types of partial engulfment but could be in error for other multiple fire impingement cases. This is a weakness in the program and is due to the present limitations of the tank car model.

The above logic was selected because of the present limitation of the tank thermal model and computing power constraints (time and memory). Future versions of DERACS could be improved by allowing for more realistic fire impingement.

k. Subroutine TANK8

Subroutine TANK8 and it associated lower level routines are a modified version of the tank thermal model TANKCAR. To create TANK8 the following modifications were performed on TANKCAR:

i) the calculations were simplified to reduce the overall computation time.

ii) the program was expanded and calculation procedures were modified so that up to eight tank cars could be considered simultaneously.

iii) chlorine and hexane were added as commodities.

iv) the main program was rewritten as a once through subroutine that is called by the DERACS main program.

v) the calculation procedure was modified to be able to account for simplified partial fire engulfment (NOTE: this feature has not been validated in any way).

The TANK8 subroutine calculates the thermal response of the tank and its lading to fire impingement. In doing so it keeps track of many important variables that are needed by other routines in the DERACS simulation such as those for release rates, plume and puff dispersion and fire and explosion effects. TANK8 also calculates the time to tank rupture or time to tank empty which ever comes first. In the event that the relief valve is open the vent mass flow is calculated and used for the purposes of plume dispersion or burning flare length calculations.

For technical details of the TANK8 the reader is directed to Reference [3] which describes the workings of TANKCAR.

l. Subroutine FIRBAL

This subroutine is used to predict the size and thermal effects of a fire ball. In the DERACS simulation it is assumed that if a propane tank has a thermally induced rupture the remaining lading will be expelled into the atmosphere and a fireball will result. Subroutine FIRBAL calculate the effective diameter and duration of the fireball and the distances to three different levels of thermal radiation to remote targets.

The effective fireball diameter is based on empirical relations for unconfined vapor cloud explosions. The thermal radiation levels are based on the assumption the fireball can be modelled as a sphere with its bottom edge at ground level. The emissive power of the fireball is also based on empirical relations. The fireball to remote target radiant exchange factors are based on those for a sphere radiating to a remote surface element oriented towards the fireball. Atmospheric attenuation of the thermal radiation is accounted for.

Technical details of FIRBALL can be found in Appendix D.5.

m. Subroutine BLEVE

This subroutine is used to predict whether or not a BLEVE will result after a tank containing propane has ruptured. The subroutine is called if a propane tank has ruptured resulting in a total loss of containment. The subroutine works by checking to see if the propane temperature at the time of rupture exceeds the superheat limit locus for the lading at atmospheric pressure. This is one theory for predicting the likelihood of a

BLEVE occurring (see Reference [4] for details). It should be noted that this is an active area of research and that new theories for predicting the probability of BLEVE's are likely. The modular nature of the present software makes future modifications very straight forward.

Further details of the approach can be found in the listing of BLEVE in Appendix D.3.

n. Subroutine BLAST

This subroutine is used to predict the consequences of an unconfined vapor cloud explosion. In DERACS it is assumed that is a propane tank ruptures catastrophically the remaining lading is expelled and a fireball results. This fireball results in thermal as well as blast loading of the surroundings. Subroutine FIRBAL is used to calculate the thermal effects while BLAST is used to calculate the blast loading effects.

Subroutine BLAST first calculates the efficiency of the unconfined vapor cloud explosion based on whether or not a BLEVE is likely. In BLAST it is assumed that if a BLEVE is not probable the explosive efficiency is 0.10 and if a BLEVE is likely the efficiency is set at 0.65. These assumptions are unsubstantiated.

BLAST then converts the remaining mass in the tank to an equivalent mass of TNT and based on this the free field overpressure can be calculated as a function of distance from the centre of the explosion. BLAST has been set up to calculate three distances based on empirically based levels of damage.

Further details of the calculation performed in BLAST can be found in Appendix D.2.

o. Subroutines PLUME and PUFF

Subroutines PLUME and PUFF are used to track continuous and instantaneous releases of heavy gases respectively. In DERACS if chlorine or propane are released due to a small puncture, or is chlorine is released due to relief valve operation then the resulting plume will be tracked. If there is a total loss of containment in a tank carrying chlorine then the resulting puff will be tracked. In all other cases of releases it is assumed that the material is burned off and is not considered important enough to track.

Both the PLUME and PUFF subroutines are based on the box type puff model. At present transient plumes are modelled as a series of puffs, each containing the mass that is released in a single time step (20 seconds). It should be noted that the present box model assumes entrainment of air occurs at the top and all around the circumference of the box and this, of course, is not the case for a plume which entrains air at the top and through the exposed sides. This error in modelling will tend to overestimate the rate at which the plume disperses and therefore underestimate the size of the danger zone. This is a weakness that should be taken care of in further work.

PLUME and PUFF generate a great deal of data including the puff radius and downwind distance as a function of time from release. The downwind distance and puff radius are stored as long as the mean concentration in the puff remains above some critical value. For chlorine this critical value is based on human response while for the propane it is based on flammability limits. Because of the volume of the data it is not stored in core but rather it is sent to disk for subsequent postprocessing. The stored data is sent to eight files named TCAR.PUF where n goes from 1 to 8 (one for each tank-car). For the case of a single instantaneous release the file will contain one record which has the time of the release followed by a series of data pairs of puff radius and downwind distance for a series of times separated by one time step. The record ends when the puff concentration drops below the critical value. For the case of continuous releases the file contains a series of records as described above, one for each puff that represents one time step worth of continuous release.

For technical details of the box puff model the reader is directed to Appendix D.11.

6. POST-PROCESSOR

6.1 Introduction

The postprocessor is the last of the DERACS modules. It is used to generate graphical and textual output from results generated in previous modules. The different forms of output are graphics on your video display terminal, graphics on your printer, and text reports. The data generated by the preprocessor, Derailment simulation and Accident simulation, are organized and processed so the data can be presented in an understandable form. A sample postprocessor screen is shown in Figure C6.1 of the User Guide (Appendix C).

The graphic display will show the following events as they progress through the simulation:

- vehicle placement
- surroundings
- compass, wind direction and speed
- ruptured vehicles (impact induced)
- ruptured vehicles (thermally induced)
- spills
- pool and torch fires
- vehicles exposed to fires
- flume tracks
- puff tracks
- explosion thermal effect zones
- explosion shock effect zones.

When viewing the accident on the computer screen, three levels are available. The three levels have the following dimensions:

10,000 x 10,000 m

1,000 x 1,000 m

200 x 200 m

Figure C6.1 in the User Guide (Appendix C) shows the menu options available to the user. These options can be invoked by selecting them with the mouse. The following is a description of the runtime options:

ZOOM IN - same as in the preprocessor

ZOOM OUT - same as in the preprocessor

Detail - This option allows you to turn on and off events as you choose. When this option is selected, a menu of events appears. By using the positive or negative value of the event, you can turn it on or off.

An event that is turned off will not be displayed on the computer screen.

Output - This option allows the user to print textual and graphical output.

Textual output will include the following:

- Time and title of derailment
- Initial conditions
- History of the events as they occurred up to the present.

Graphical output will be a snapshot of the screen in its present state.

Note: Events turned off through detail will not be printed.

Key - This will display the key page.

START - This starts the postprocessor displaying output at computer speed.
Computer speed is a combination of disk and clock speed. Generally, each picture will be drawn in 5 → 10 seconds.

To stop the postprocessor and zoom or produce output, you press the mouse and hold until the current picture has completely been drawn.

EXIT - This will exit the postprocessor.

6.2 Description of Subroutines

The following is a description of the major subroutines in the postprocessor. The format is the same as the preprocessor.

PPOST - This function is the main loop of the postprocessor. This function loops in the menu area until a command is chosen.

ICNUM -
(I) Command Number
CLICK -
(L) Mouse Condition

SUBZOM - Same as in the preprocessor.

DETAIL - This function determines the level of detail to be shown when displaying events.

IDETAL -
(I) Array contining the event markers. If an event is set to 0, it is not displayed.

PRT - This function takes a snapshot of the screen and dumps it on your dot matrix printer.

KEY - Same as in the preprocessor.

START - This function starts the postprocessor reading data and displaying it on the screen. After each display, the mouse is polled for condition. If the mouse button is being pressed, the function ends and control is returned to the main loop.

OUTFLG - (L) Exit flag set to true if simulation is complete

IGLBTM - (I) Global time of simulation. Incremented each time through the loop.

UPDAT2 - This function updates the status area.

DEWIC2 - This function redraws the menu area.

NEXTST - This function reads in the next set of DERACS simulation data and records the vital information needed to generate a final text summary. The screen is then redrawn showing all updated events.

DFIRE - This function displays any fires.

DSPLL - This function displays any spills.

DBLAST - This function displays any blasts.

DPUFF - This function displays any puffs.

DPLM - This function displays any plumes.

DFLAR - This function displays any flares.

PUTPUT - This function dumps the text summary to the given output devices.

7.0 SAMPLE SIMULATION

This chapter presents the results of two sample simulations that were performed using the prototype DERACS program. The sample simulations were loosely based on the Mississauga rail accident of November 1979. The results about to be presented have not been validated in any way and should not be considered as accurate representations of the Mississauga accident. They have been generated for discussion purposes only so that future applications of the DERACS program could be illustrated.

7.1 The Mississauga Derailment

The Mississauga derailment occurred on November 10, 1979 and involved a train consisting of 106 cars. The derailment was initiated by the 33rd car in the consist and in total 24 cars were derailed. Of these 24, 21 were tank cars with 19 of these carrying dangerous goods including chlorine, propane, toluene, caustic soda, and styrene. It was not determined for certain what the train speed was at the time of the derailment but witnesses and estimates place is somewhere between 70 and 100 km/hr. Figure 7.1 presents a sketch of the derailment scene from Reference [5]. Table 7.1 presents a listing of the derailed cars and their contents.

The 7th derailed car carried chlorine and this car suffered an impact or ferric chloride reaction inducted puncture .76 m in diameter. As a result of this rupture chlorine was released into the environment. Other cars sustained impact induced ruptures resulting in the release of flammable liquids. Of the derailed cars, those cars with bottom fittings (all of the type 111 cars except car #24) had the fittings sheared off during the accident and as a result their contents were released.

From the photographs shown in Reference [5], an intense fire was located at or near the 15th derailed car which was containing toluene. This fire and others were likely the cause of the thermal ruptures of the 8th, 12th and 13th derailed cars which carried propane. The explosions of these propane cars resulted in extensive damage to neighbouring properties. At least one of the explosions occurred very early on in the event which leads one to conclude that it was an impact induced rupture. The explosions also resulted in large pieces of the tanks being thrown large distances (over 2 km).

The above facts were used as input to the sample simulations performed using DERACS. Because of certain limitations of DERACS some simplifying assumptions were required.

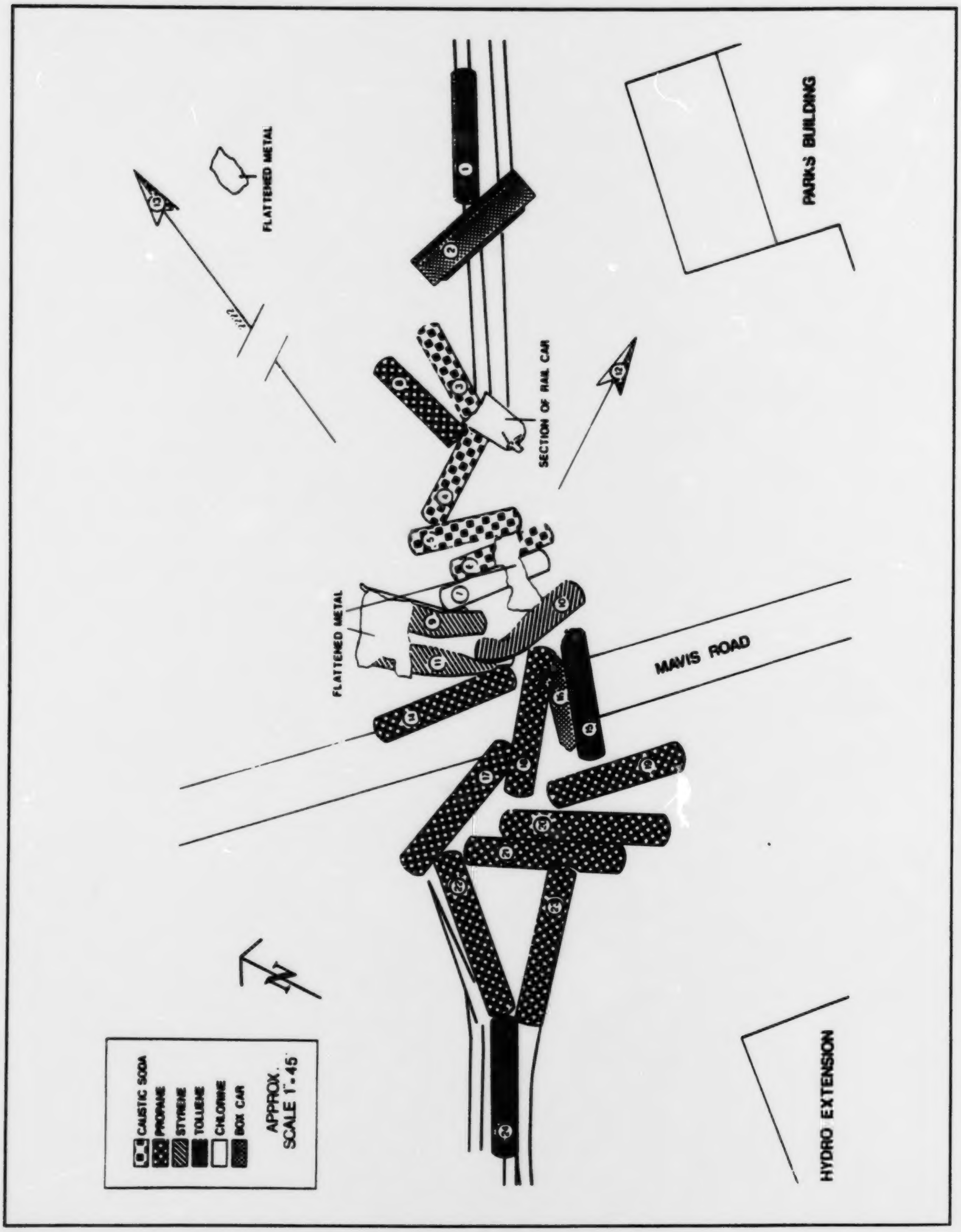

PARKS BUILDING
FLATTENED METAL
SECTION OF RAIL CAR
FLATTENED METAL
MAVIS ROAD
HYDRO EXTENSION
N
CAUSTIC SODA
PROPANE
STYRENE
TOLUENE
CHLORINE
BOX CAR
APPROX.
SCALE 1":45'

Figure 7.1 Mississauga Derailment Site (From Reference [5])

Table 7.1 Summary of Derailed Cars and Their Contents

Derailed car Number	Contents	Insulated	Punctured
1	toluene		
2	box car		
3	caustic soda		
4	caustic soda		
5	caustic soda		
6	caustic soda		
7	chlorine	Yes	Yes
8	propane	Yes	Explosion
9	styrene		
10	styrene		
11	styrene		
12	propane	No	Explosion
13	propane	No	Explosion
14	propane		
15	toluene		Yes
16	toluene		?
17	propane		
18	propane		
19	propane		
20	propane		
21	propane		
22	propane		
23	propane		
24	toluene		

7.2 DERACS Simulation Input

At present DERACS is capable of simulating the derailment mechanics of up to 40 cars and of these 8 can be tank cars carrying chlorine, propane or hexane. In order to perform a simulation of the Mississauga derailment it was necessary to transform the actual event as described above into a similar event that fits within the operating constraints of DERACS.

Surroundings:

The Mississauga derailment occurred at Mavis Road in the city of Mississauga. An approximate map of the surrounding area (10 km by 10 km) was entered using the DERACS preprocessor and is shown in Figure 7.2. This map does not contain all of the details of the area but does show approximately areas of water, forest and inhabited areas as well as roads and rail lines. The DERACS preprocessor can be used to enter far more detail than is shown in the global view of Figure 7.2.

Atmospheric Conditions:

The wind speed was reported in Reference [5] to be 28 km/hr. This was entered along with an assumed atmospheric pressure of 101325 Pa and wind direction coming from the North West. An assumed air temperature of 10 deg C was used.

Train Consist:

For the derailment simulation a train consist of 36 cars was used including 8 tank cars. The consist was made up as shown in Table 7.2. For all tank cars it was assumed that they were filled to 90% capacity and their initial temperature was 10 deg C.

Derailment Point:

The derailment point was located at the intersection of Mavis Road and the rail line. The rail line was assumed to be straight track at the derailment point. The first truck of simulation car #4 was assumed to be the derailment initiator.

Derailment Speed:

As stated in Reference [5] the train speed at the point of derailment was not known for certain. As a result a number of simulation runs were performed using different train speeds. It was found that a train speed of 100 km/hr gave a reasonable derailment scene with very good correlation to the actual derailment scene. In this case it was considered important to be able to predict the correct number of derailed cars, and the total area covered by the derailed vehicles.

It should be noted that the derailment mechanics simulation software is very sensitive to train speed. This sensitivity is due to the use of loose tolerances in the numerical solution procedure. The loose tolerances were necessary so that the program could run in a reasonable length of time on the IBM/AT machine.

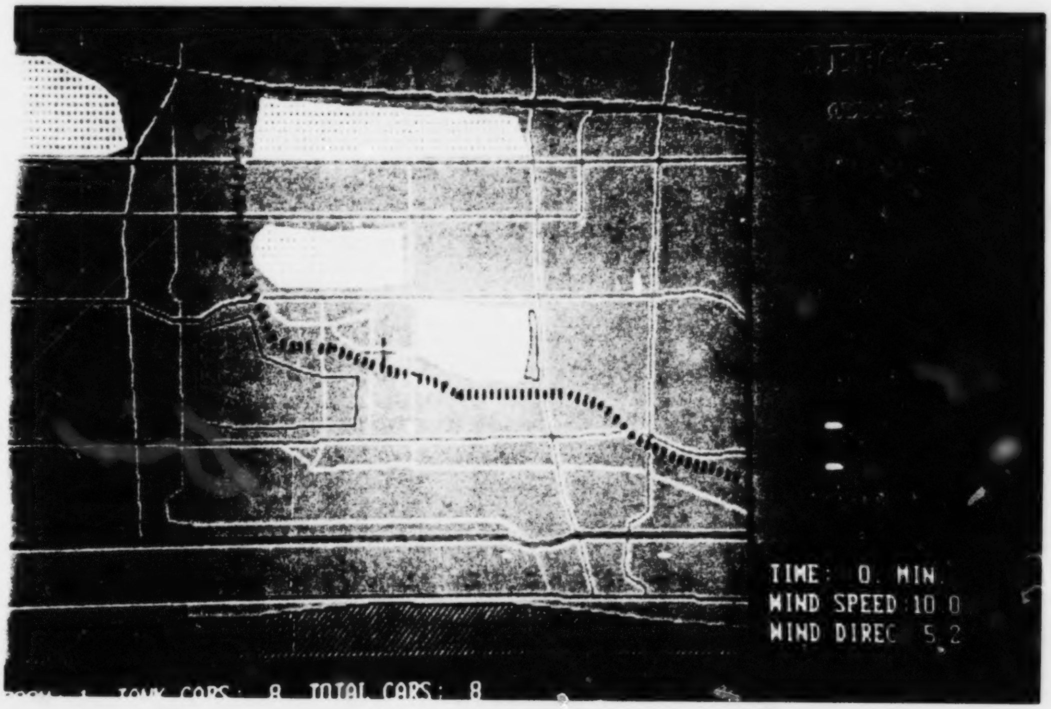

TIME: 0. MIN.
WIND SPEED 10.0
WIND DIREC 5.2
TANK CARS: 8 TOTAL CARS: 8

Figure 7.2 Approximate Map of Derailment Site (From Reference [5])

Table 7.2 Summary of Consist Used in DERACS Simulation of Mississauga Derailment

Derailment car Number	Simulation	Simulation Car Number	Protection Car Type
-	1	-	
1	2	-	
2	3	-	
3	4	-	
4	5	-	
5	6	-	
6	7	-	
7	8	chlorine	insulated
8	9	propane	insulated
9	10	-	
10	11	-	
11	12	-	
12	13	propane	
13	14	propane	
14	15	propane	
15	16	hexane	
16	17	hexane	
17	18	propane	
18	19	-	
19	20	-	
20	21	-	
21	22	-	
22	23	-	
23	24	-	
24	25	-	
-	26	-	
-	27	-	
-	28	-	
-	29	-	
-	30	-	
-	31	-	
-	32	-	
-	33	-	
-	34	-	
-	35	-	
-	36	-	

Vehicle Placement and Assumed Ruptures:

With DERACS it is possible to simulate the derailment mechanics to predict the vehicle placement and impact induced ruptures or these conditions can be entered using the DERACS preprocessor. For the purposes of the sample simulations both approaches were used. Figure 7.3 presents the assumed vehicle placement as based on Reference [5]. For impact induced ruptures it was assumed that car 8 (chlorine) was punctured and cars 9 (propane), 16 and 17 (hexane) sustained total loss of containment.

7.3 Sample Simulation Results

As stated above two sample simulations were performed. The first case included a simulation of the derailment mechanics where the second case had the vehicle placement and ruptures as input. The results of these two simulations will be presented separately.

7.3.1 Simulation Case 1 - Derailment Mechanics Used to Predict Vehicle Placement

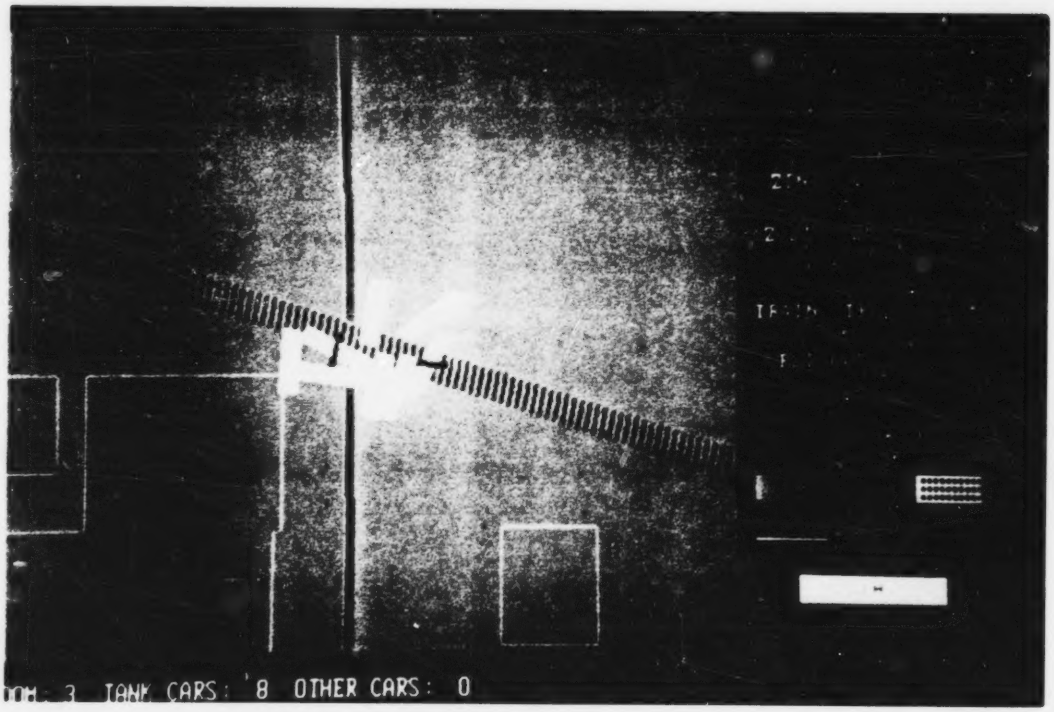
This case involved the simulation of the derailment mechanics as well as the simulation of the resulting consequences.

Figure 7.4 shows a plan view of the derailment scene as predicted by the DERAIL module of DERACS. The important features to note are that 24 cars have left the track exactly as in the actual Mississauga derailment. The total length of the track involved is 140 m which is in very good agreement with the actual accident.

As a result of the simulation it was predicted that three impact punctures resulted including two of the propane cars and one of the hexane cars. This is indicated in the Figure. Note that this simulation cannot account for the loss of bottom fittings which had a very significant effect in the Mississauga accident. The puncture of the hexane car results in a spill of flammable liquid and the propane car punctures result in heavy gas plumes. (NOTE: the present version of DERACS handles a leak of propane as a heavy gas dispersion problem and not as a local fire or explosion. If a propane tank suffers a total loss of containment then it is assumed that an explosion will result. These assumptions can be changed in later versions of DERACS).

Following the derailment mechanics simulations the DERACS main simulation program was used. The results of this simulation can be summarized as follows. The hexane spill resulted in a fire that engulfed one unpunctured propane car. Two plumes issued from the punctured propane cars. As the simulation progressed the fire slowly burned itself out and as it did the pool fire diameter decreased in size until the fire impinged tanks were no longer impinged by fire. As a result no thermal ruptures were predicted. The heavy gas plumes dispersed downwind as expected.

For this simulation the most significant event was the release of the propane gas as both an explosion and toxic hazard. Figures 7.5 to 7.9 show the derailment scene at various times during the simulation.



OOM: 3 TANK CARS: 8 OTHER CARS: 0

Figure 7.3 Input Vehicle Placement and Location of Ruptures

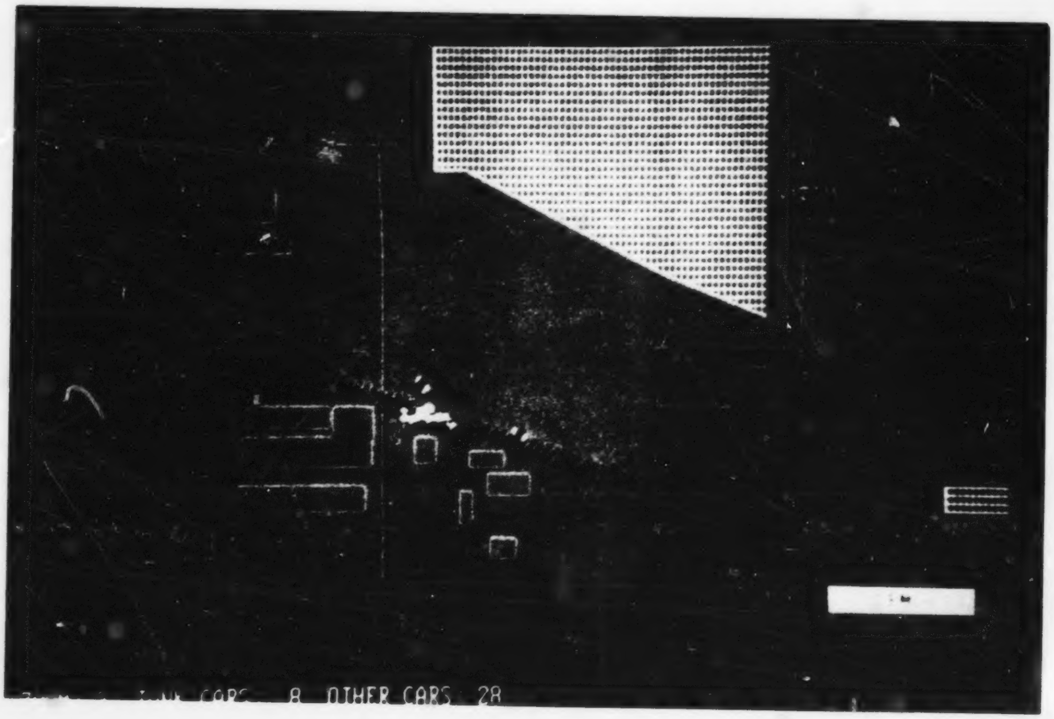

CARS 8 OTHER CARS 28

Figure 7.4a Predicted Vehicle Placement and Rupture Locations (Zoom Level 2)

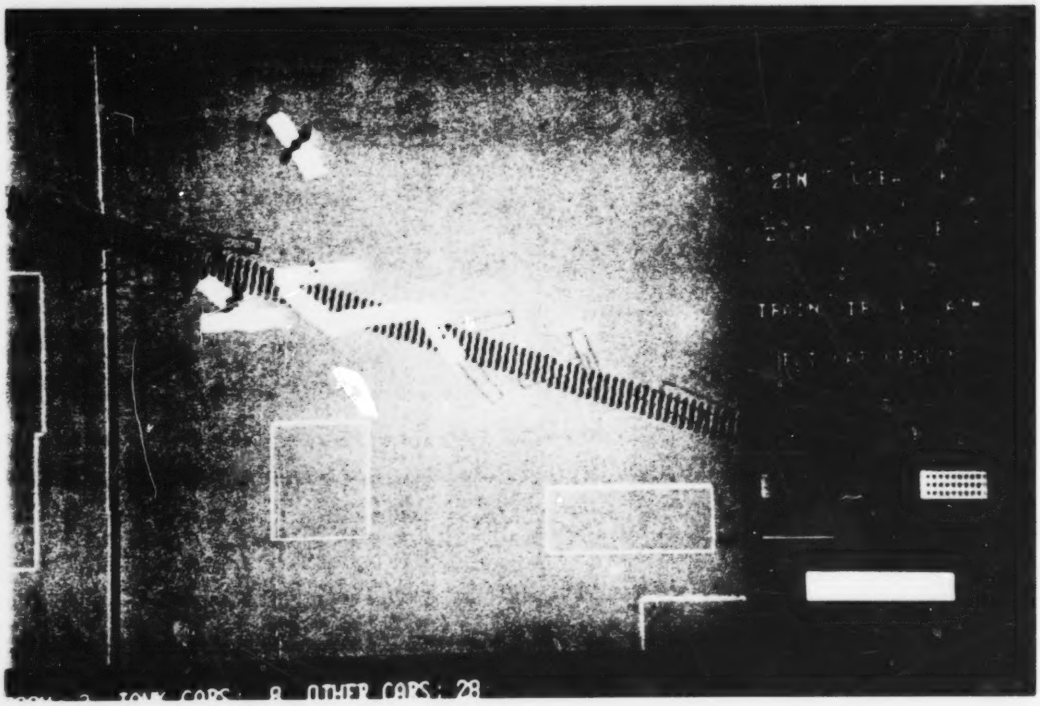

TANK CARS: 8 OTHER CARS: 28

Figure 7.4b Predicted Vehicle Placement and Rupture Locattions (Zoom Level 3)

7.3.2 Case 2 - Input of Vehicle Placement and Simulation of Resulting Consequences

In this sample simulation the vehicle placement was provided as input along with the location of punctures. Figure 7.3 showed the assumed placement and the assumed punctures were described in an earlier section.

The DERACS simulation software was used to simulate the consequences of the assumed initial conditions. The results of this simulation can be summarized as follows.

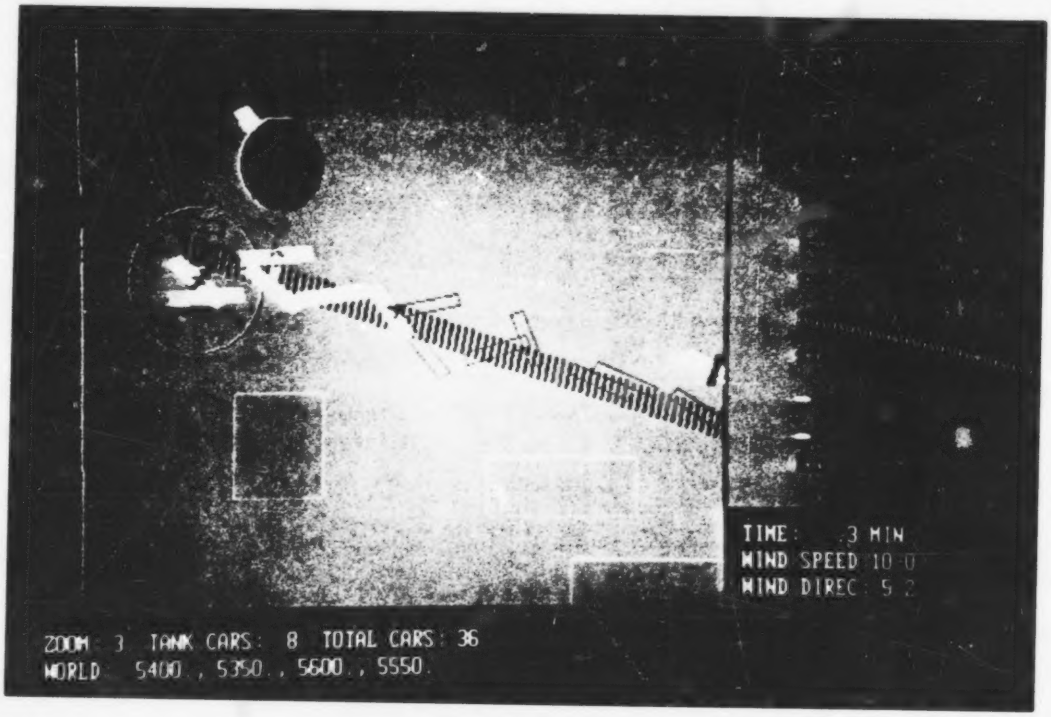
The assumed total loss of containment of the propane car resulted in an explosion and fire ball (not a BLEVE). The resulting thermal and blast loads resulted in the destruction of nearby buildings. The puncture of the chlorine car resulted in the release of chlorine gas into the environment. This plume drifted downwind for some distance at lethal concentrations. The total loss of containment of the hexane car resulted in a large pool fire that initially engulfed or partially engulfed several propane cars. As the fire burned itself out the pool fire diameter decreased and slowly the number of tank cars impinged by fire were reduced. However at 24 minutes into the simulation one of the uninsulated tank cars experienced a BLEVE. This BLEVE caused thermal and blast loads to a large area. However, damage from the BLEVE added little to the damage done by the fire ball and explosion of the first punctured propane tank car.

Figures 7.10 to 7.14 present view of the simulation at different times.

7.4 Discussion

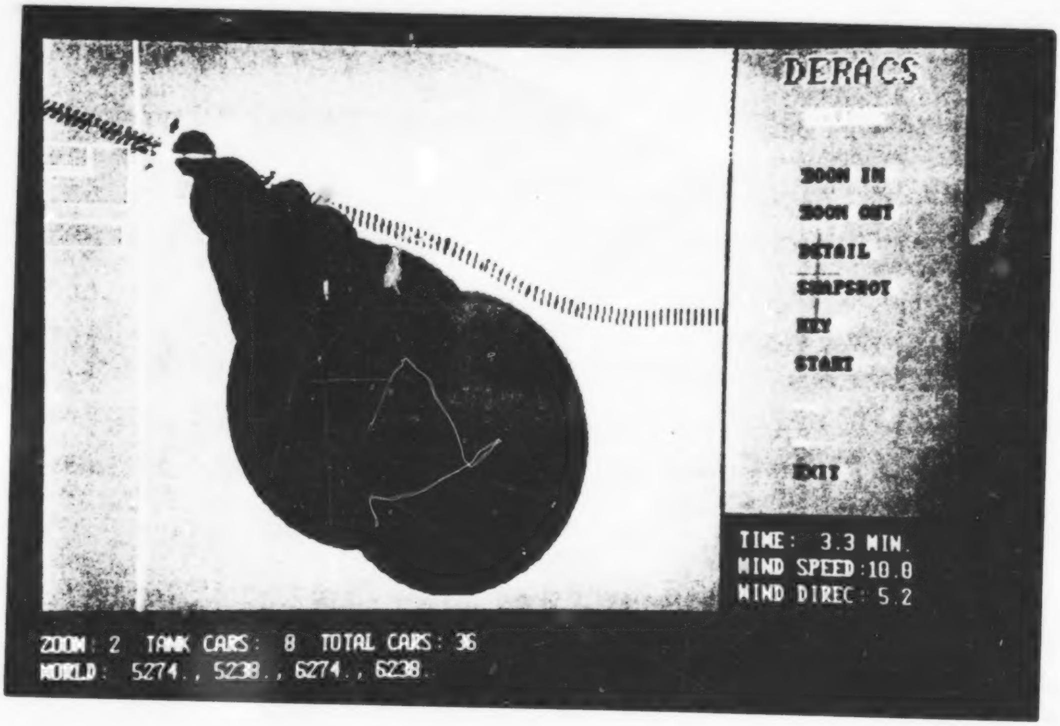
The previous examples show the capabilities of the prototype version of DERACS. As expected there are differences between reality and simulation. However, considering the limitations of the present version of DERACS the results are very promising.

The derailment mechanics simulation was very accurate at predicting the total number of cars derailed and the size of the area affected by the derailed vehicles. This of course was a function of the assumed train speed at the time of the derailment. The derailment mechanics routines were not as successful at predicting the location and numbers of punctures and this in turn resulted in predicted consequences that differed considerably from those observed in the Mississauga derailment. That is, there was no predicted release of chlorine and no predicted thermal ruptures. It should be noted that the present impact rupture routines are very crude so that they could run quickly on the IBM/AT machine. The impact rupture routines require further development if more accurate and realistic results are to be obtained.



```
TIME:    .3 MIN
WIND SPEED 10.0
WIND DIREC: 5.2
ZOOM: 3  TANK CARS:  8  TOTAL CARS: 36
WORLD:  5400., 5350., 5600., 5550.
```

Figure 7.5 Case 1 Simulation Results at 0.3 Min.


DERACS
ZOOM IN
ZOOM OUT
DETAIL
SNAPSHOT
KEY
START
EXIT

TIME: 3.3 MIN.
WIND SPEED:10.0
WIND DIREC: 5.2

ZOOM: 2 TANK CARS: 8 TOTAL CARS: 36
WORLD: 5274., 5238., 6274., 6238.

Figure 7.6 Case 1 Simulation Results At 3.3 Minutes

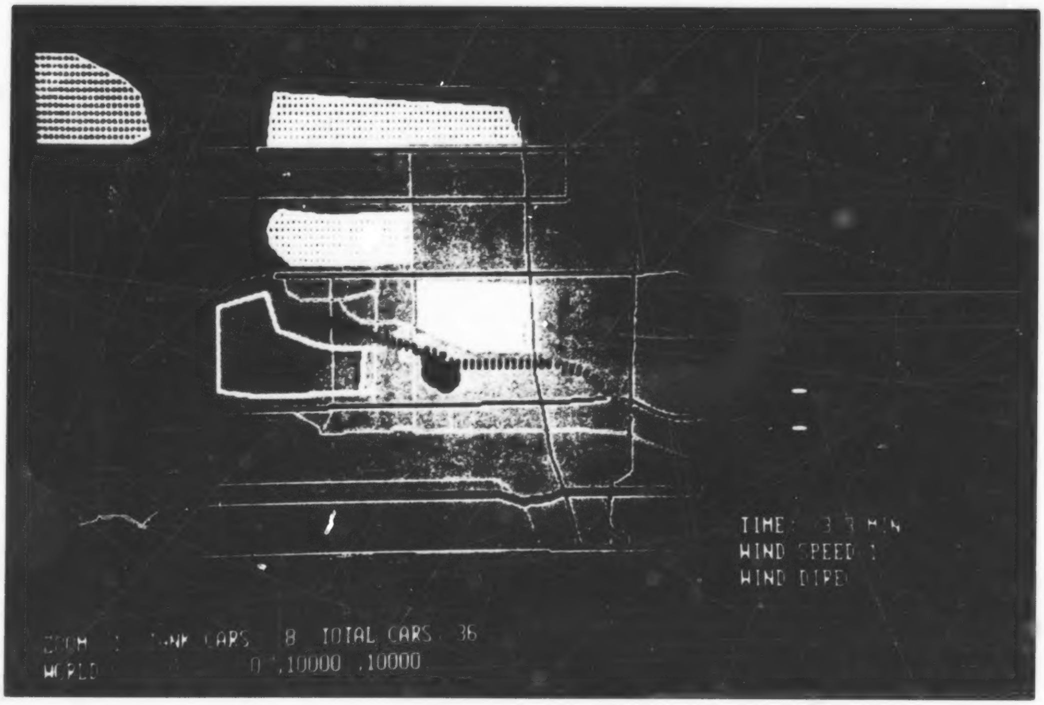

TIME: 3.3 MIN
WIND SPEED 1
WIND DIR
ZOOM 1 TANK CARS 8 TOTAL CARS 36
WORLD 0 ,10000 ,10000

Figure 7.7 Case 1 Simulation Results at 3.3 Minutes

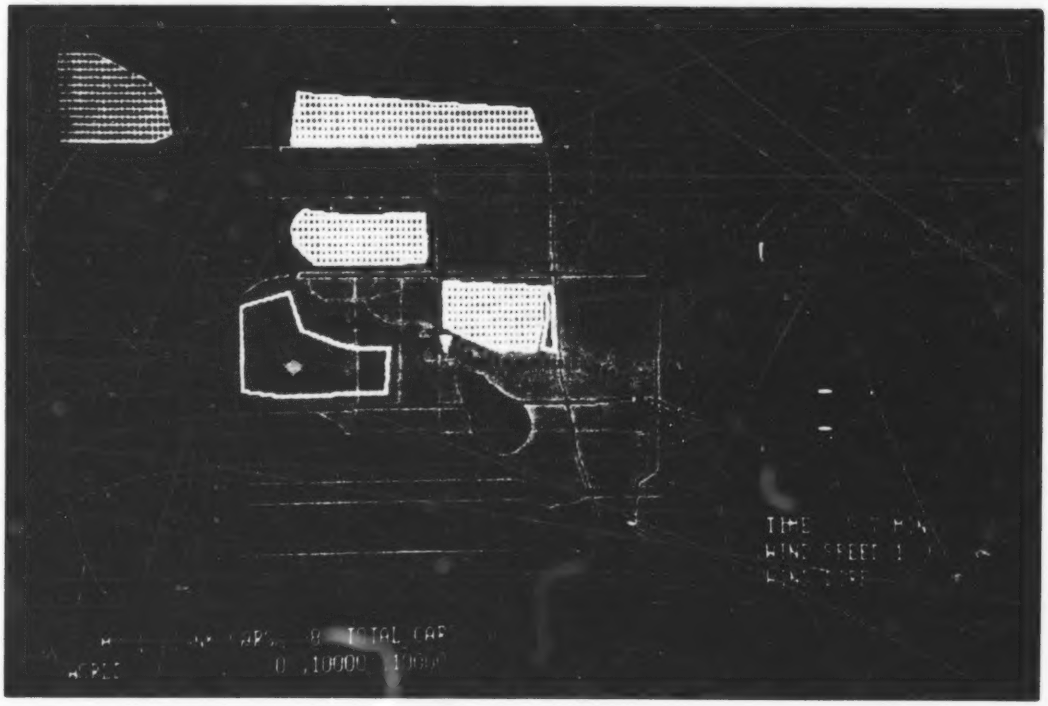
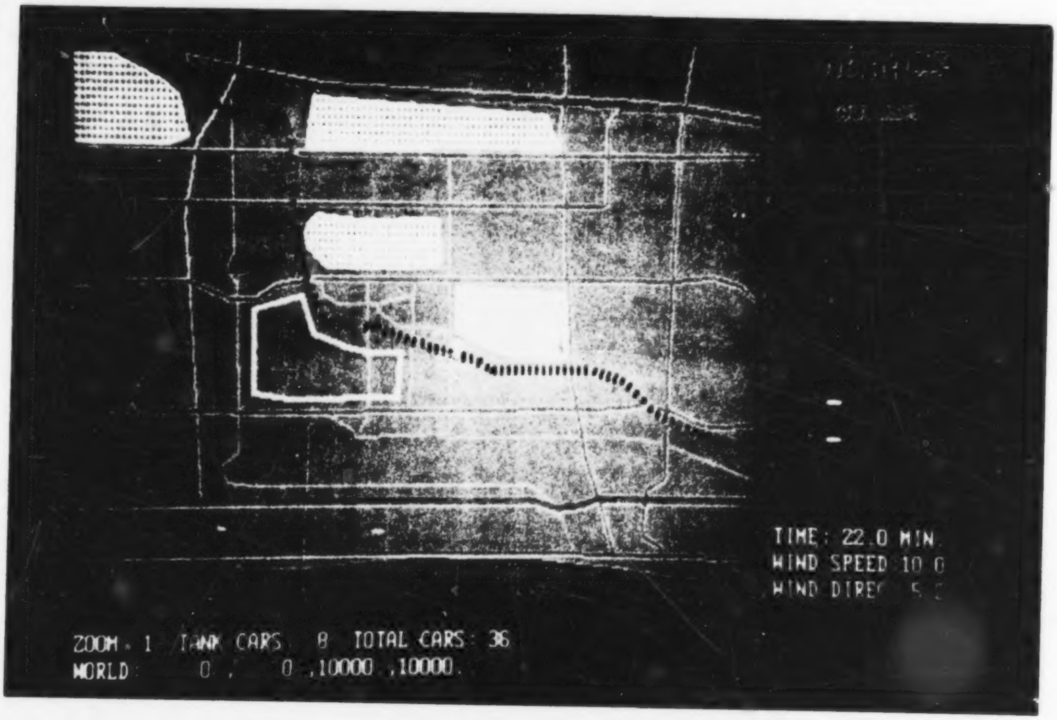

TIME
WIND SPEED
TOTAL CAR
0 .10000

Figure 7.8 Case 1 Simulation Results at 9.7 min.



```
TIME: 22.0 MIN.
WIND SPEED 10.0
WIND DIREC 5.[illegible]

ZOOM 1 TANK CARS 8 TOTAL CARS: 36
WORLD:  0 ,  0 ,10000 ,10000.
```

Figure 7.9 Case 1 Simulation Results At 22 Minutes

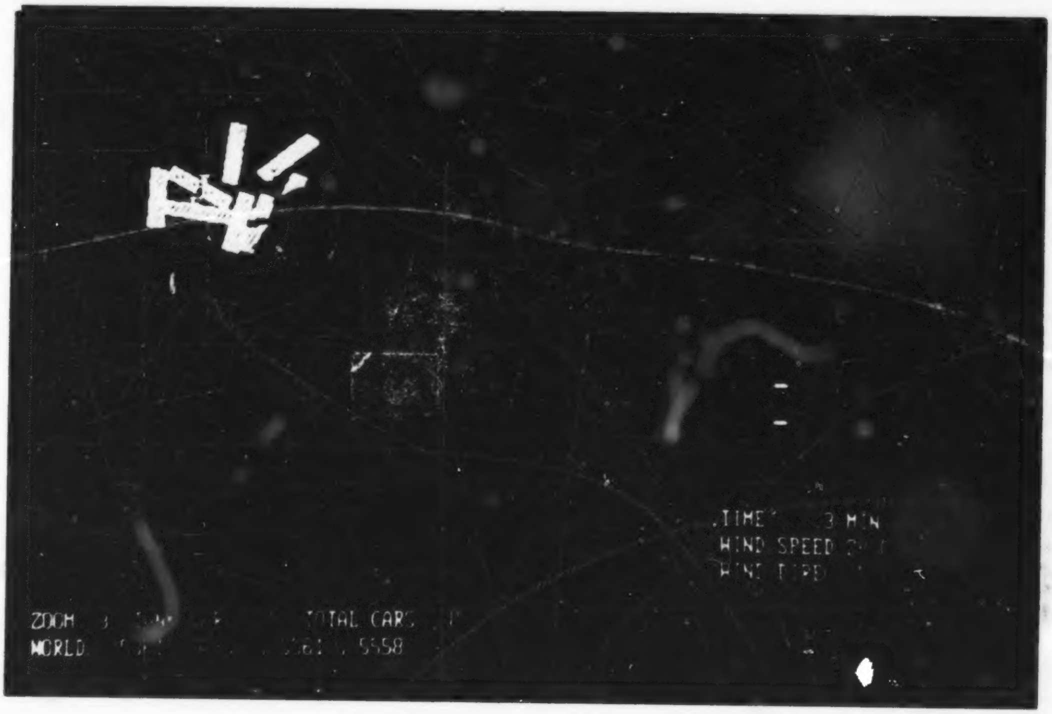

ZOOM 3
TOTAL CARS
WORLD
5558
TIME 3 MIN
WIND SPEED
WIND DIR

Figure 7.10 Case 2 Simulation Results At 0.3 Min.

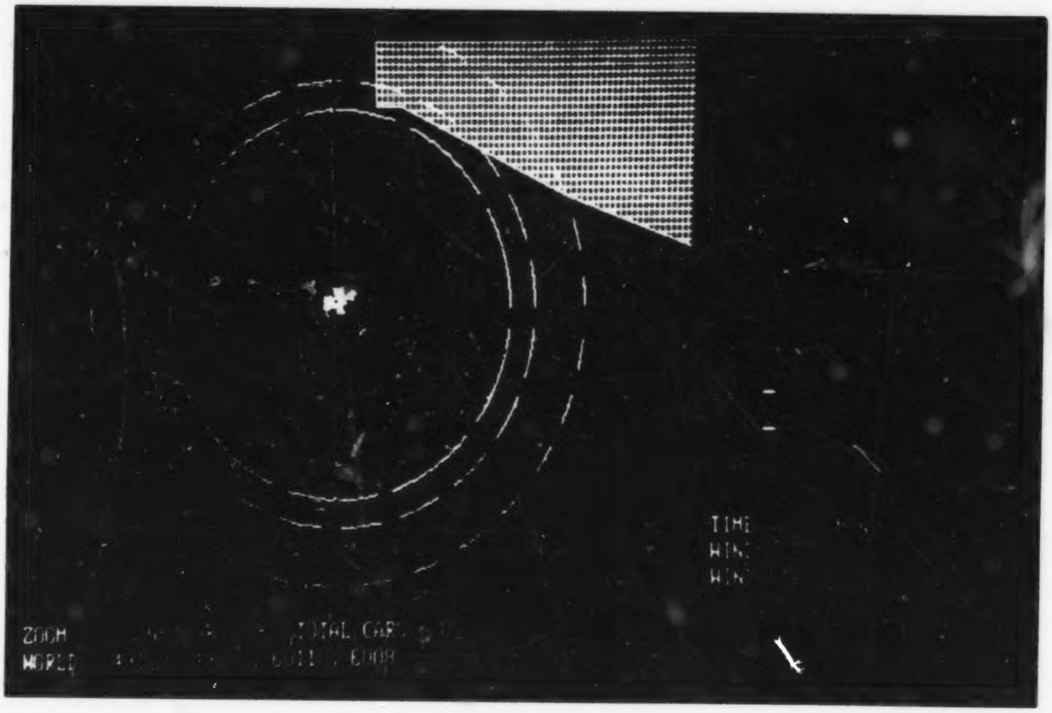
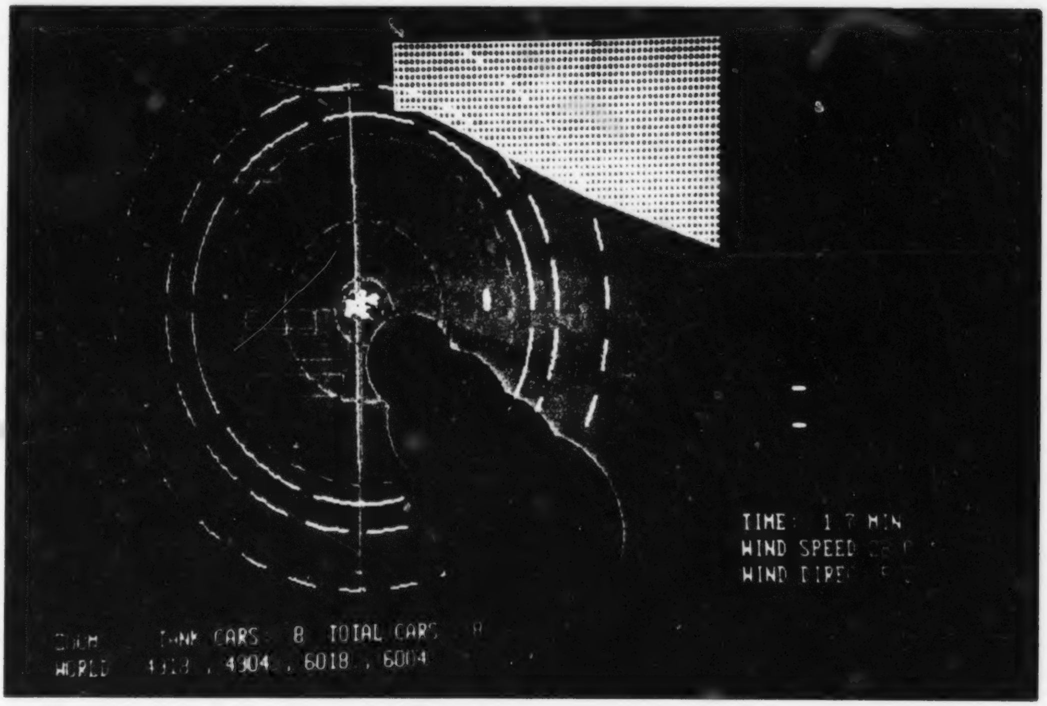

TIME
MIN
MIN
ZOOM
TOTAL CARS
WORLD
6008

Figure 7.11 Case 2 Simulation Results At 0.3 Min.



TIME: 1.7 MIN
WIND SPEED [illegible]
WIND DIREC [illegible]
ZOOM TANK CARS: 8 TOTAL CARS: 8
WORLD: 4318 , 4304 , 6018 , 6004

Figure 7.12 Case 2 Simulation Results At 1.7 Minutes

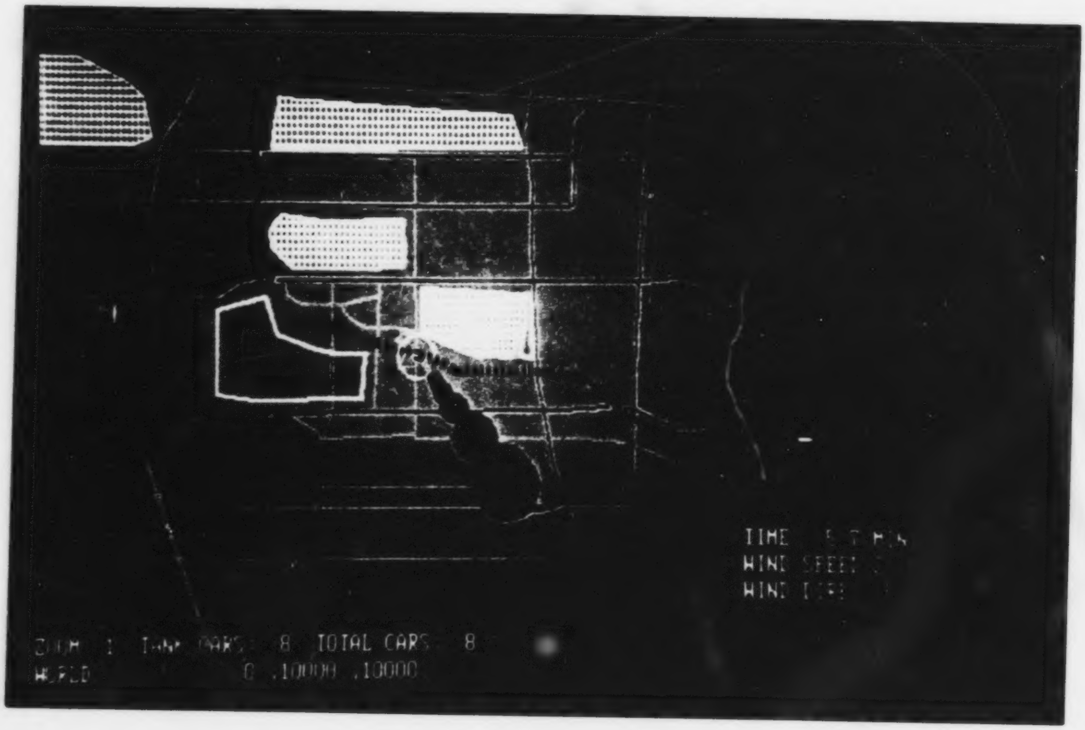

TIME 5.7 MIN
WIND SPEED
WIND DIR
ZOOM 1 TANK CARS 8 TOTAL CARS 8
WORLD 0 .10000 .10000

Figure 7.13 Case 2 Simulation Results At 5.7 Min.

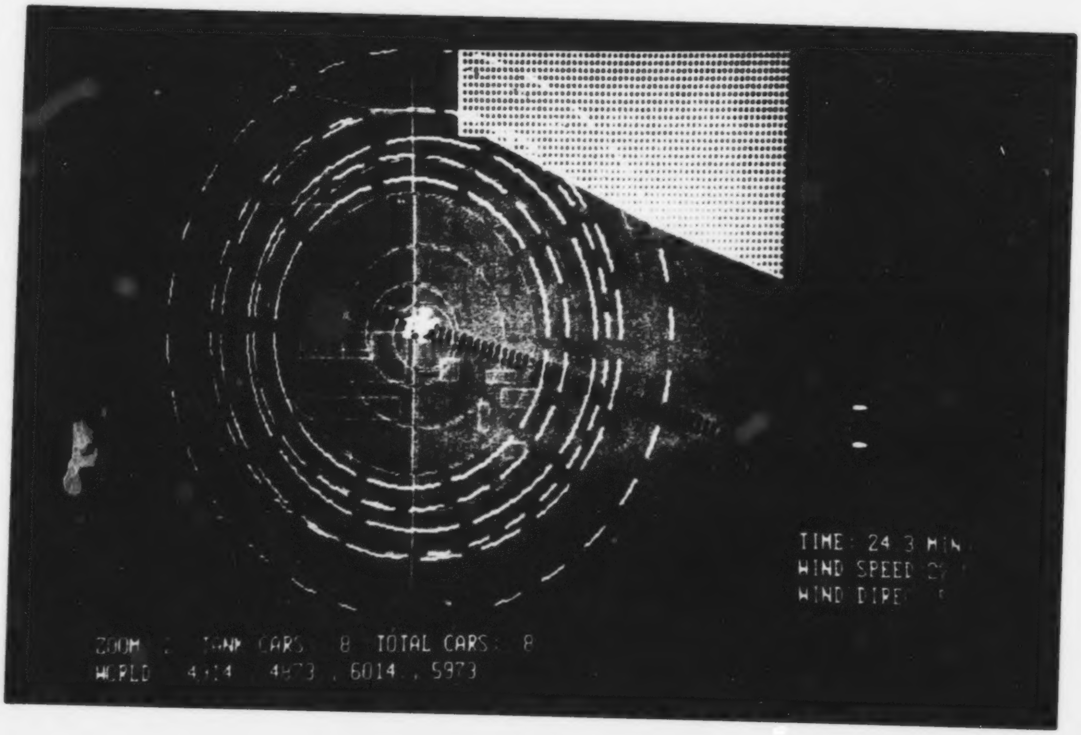

TIME: 24.3 MIN
WIND SPEED 2
WIND DIRE
ZOOM 2 TANK CARS 8 TOTAL CARS: 8
WORLD 4 14 4873 , 6014 , 5973

Figure 7.14 Case 2 Simulation Results At 24.3 Min.

For the case where the vehicle placement and puncture locations were entered as input the simulation was much more realistic. This of course is an obvious result because adding more information as input naturally results in better output. It should be remembered however that only the vehicle placement and locations of punctures were input. No data pertaining to release rates, or pool fire sizes were input. All of these variables were handled using the DERACS submodels.

These sample simulations have illustrated that DERACS can be used to study derailment accidents in an informative manner, and therefore shows promise as a tool for accident reconstruction. However, it appears that as a predictive tool DERACS gives better results if the derailment mechanics routines are bypassed. This bypass is practical, say, for a response team at the site of an accident where the location of vehicles and punctures is known. In such a situation DERACS could be run to give a possible view of what to expect for the next few hours in the accident. On the other hand, the results of the full simulation which included the derailment mechanics was not unrealistic even though they did not agree well with the Mississauga event. Therefore this leads one to conclude that DERACS can be used as a derailment accident simulator. Such a simulator could be valuable for the purpose of response team training, for example.

8. FUTURE DEVELOPMENT

The present project was to develop a prototype software package for simulating derailment accidents. If this prototype proves useful it could then be further developed into a comprehensive software package for simulating rail and highway accidents involving tankers carrying dangerous commodities. Such a package could prove useful for risk studies, response training, accident resconstruction and possibly as a tool to be used on site at an accident scene.

The following sections discuss the strengths and the weaknesses of the present version of DERACS. Possible applications are discussed and recommendations are made as to the possible next phase of development of the program.

8.1 Program Weaknesses

The present version of DERACS is a prototype and as such is very limited in its capabilities. The program presently contains many simplifying assumptions which in some cases are not supported by existing data. The program is not yet a truly accurate simulation of all the significant processes that can take place in a derailment accident.

The following sections contain brief discussions of weaknesses and how they can be improved. Table 8.1 contains a summary of the identified weaknesses.

8.1.1 General Weaknesses

The DERACS simulation software can be enhanced or further developed in many areas. However certain areas have been identified as being significant in terms of weakening the overall predictive capabilities of DERACS. First area of general weaknesses involves the small number of commodities that the DERACS model can presently consider. DERACS can now handle propane, chlorine and hexane as possible commodities carried by the tank cars in the simulation. There are many other dangerous commodities that are presently transported by tankers. DERACS should be developed to include many more of the currently identified dangerous commodities.

Another weakness is associated with how punctured vehicles are handled. If, at the beginning of a simulation, a tank is ruptured, then the release calculations are performed assuming the tank is in ambient conditions. However, it is possible that a ruptured tank is impinged by fire. This fire impingement would significantly effect the rate at which material is released. Future versions of DERACS should include modifications such that the tank thermal model carries out calculations for fire impinged tanks with small punctures. At present, CONREL (derived from LEAKR (Fingas et al)) carries out the release calculations and it presently does not account for fire impingement.

Table 8.1 Summary of DERACS Simulation Weaknesses

AREA	WEAKNESS	SOLUTION
General	- Release rates from punctured tanks does not include effects of fire impingement.	- The tank thermal model should be enhanced to account for punctures.
	- Few commodities	- Add commodities
	- Ground is flat	- Add terrain
	- Specialized topics (eg. ferric chloride reaction between chlorine and steel and tank rocketing)	- Add additional models
	- Long simulation run times	- Optimize programs or use faster computers
Derailment Mechanics	- Roll degree of freedom is weak	- Improve calculations
	- Impact rupture analysis is simplistic	- Improve calculations
	- Maximum number of cars limited to 40	- Improve integration scheme to handle more cars
	- Vehicle types are not handled in detail (tank-cars and are not differentiated by type, i.e. 111, 112, 115, etc.)	- Increased level of detail for vehicle types
	- No grade effects of freedom	- Add pitch and elevation as degrees
Tanker Fire Impingement software	- Single fire case	- Upgrade TANKCAR overhead
	- Fire case based on single worst impingement not on sum	- As above
	- Only eight tank-cars	- Expand to memory limits

Table 8.1 Summary of DERACS Simulation Weaknesses (cont'd)

AREA	WEAKNESS	SOLUTION
Releases, Plumes, Puffs	- Box puff model used for plumes	- Replace with box plume model
	- Terrain not accounted for	- Upgrade plume and puff models
	- Only round ruptures	- Add other shapes
Explosions	- BLEVE predictions simplistic	- Upgrade as data becomes available
to nearby tanks	- Blast waves and fireball thermal effects do not cause tanks	- Include calculation of blast and thermal effects on remote ruptures

At present the DERACS simulation takes place assuming that the ground is perfectly flat. This makes a number of calculations difficult. Although the pre-processor allows for the input of elevation contours these contours are not used in any way in the simulation models. The elevation contours are allowed purely for visual representation of the surroundings by the post-processor. With a flat surrounding it is not possible to simulate the effect that terrain has on plume and puff dispersion, spill pool diameter and shape and vehicle placement subsequent to a derailment. A major future task would be to include the effects of terrain in the DERACS simulation. This would require the upgrade of many of the DERACS subroutines including the derailment mechanics simulation and the plume and puff dispersion subroutines.

As well as those noted above there are many specialized topics that should be addressed in such a derailment accident simulation program. Special topics such as tank rocketing as a result of rupture. As well, materials related models such as the ferric chloride reaction between chlorine and steel could also be included to add realism in the simulation. It is believed by some that this reaction played a significant role in the Missassauga accident of 1979. A literature search should be carried out to identify some of the more important specialized topics that could be included in the DERACS simulation. Future work would consider some of these special topics.

8.1.2 Derailment Mechanics

At present, the derailment mechanics simulation contains several major simplifications so that the simulation can run in a reasonable length of time on an IBM AT compatible computer. The roll degree of freedom and the impact rupture analysis are both areas that could use further refinement. They are presently functional but include many unsubstantied assumptions.

In addition to those areas stated above, the derailment mechanics portion of the simulation would benefit from the addition of track side grade effects. This, of course, would require the addition of another two degrees of freedom (elevation, pitch).

With the above further developments, there is no question that the run time would be significantly longer. It is recommended that if the above features are added, a more powerful computer be used. An improved time integration scheme is also needed if the total number of cars is to be increased from the present limit of 40.

At present, DERAIL does not consider the vehicle types in detail. Tank-cars are not differentiated by type (111, 112, 115, etc.). This detail should be added to the simulation.

8.1.3 Tanker Fire Impingement

Since the tank car thermal model plays a major role in the overall DERACS simulation any weaknesses in this model are significant in relation to the overall accident simulation. At present there are several significant weaknesses with the tank car thermal model.

At present the fire impingement case is based on a rather simplistic model that can only handle one type of fire impingement per tank per simulation. In other words if at the beginning of the simulation a tanker is exposed to a torch type fire the fire type can not switch to engulfing type fire later on during the simulation. This is because of the overhead software in the tank car thermal model. When a fire type is selected overhead software in the tank car thermal model sets up the various matrices that account for the heat transfer in the tank wall and the development of the wall temperatures. At present when an engulfing fire case is selected tank symmetry about a vertical axis is used to limit the size of the matrix. However, in the case of torch fire impingement this vertical symmetry does not exist and therefore matrices are set up to account for both sides of the tank. These routines that set up the various matrices would have to be modified to allow for fire impingement cases switching modes during a simulation.

During the DERACS simulation a tank may be exposed to many different fire sources. These fire sources may be from torches from various tanks or from a number of different pool fires occurring simultaneously. At present the fire case in the DERACS simulation is based on the single worst case fire impingement. A more realistic approach would be to account for the combined effects of the various fire sources. At present this cannot be accounted for by the tank car thermal model overhead software. Future work should be addressed to modifying the overhead software in the tank car thermal model to allow for combined effects of multiple fire impingement.

At present DERACS can only handle 8 tank cars in a simulation. This of course is very limiting and future versions of DERACS should be upgraded to allow for more tank cars. This can be done by expanding the various arrays and matrices to memory limits. It may be necessary to use a more powerful computer if the number of tanks is increased significantly. More tank cars will also require more computation time.

8.1.4 Releases, Plumes, Puffs

When a tank has total loss of containment it is assumed in the DERACS simulation that 100% of the remaining lading is expelled into the atmosphere in a single puff. In reality this does not happen. In fact, only a portion of the remaining liquid is expelled as a vapor cloud. Some of the lading is expelled in the form of aerosols and the rest of the liquid lading remains in the broken open tank where it boils off over some period of time. The boil off rate is dependent on the surroundings, conditions and whether or not a fire is present. These effects are not presently accounted for and represent a significant weakness in the way that DERACS simulates the releases of commodities after a total loss of containment accident. Future DERACS versions should include more sophisticated models to account for these effects.

The releases from ruptured tank cars are tracked using a box type puff model which simulates plumes as a continuous series of puffs. At present the box model used is not appropriate for simulating continuous releases and therefore future work should include the replacement of the present box puff model with a proper box plume model for plume tracking.

As mentioned earlier in this chapter the addition of terrain to the overall DERACS simulation would significantly enhance the realism of the simulation. This however would be a major upgrade and would require significant effort. Plume and puff models exist presently that can account for the effects of terrain however if they were used in the DERACS simulation it would be necessary to use a significantly more powerful computer to carry out the calculations in a reasonable length of time.

In addition to tracking puffs and plumes there are general weaknesses with the liquid and vapor release calculations. For example one weakness is the limitation that only round inward punctures are allowed for during the derailment impact simulation. It is well known that impact punctures of various shapes can occur during vehicle impact and these include long gashes or long rectangular inward punctures. Future versions of DERACS should allow for other shapes of ruptures as a result of vehicle impact.

8.1.5 Explosions

The effects of explosions (blast overpressure and thermal radiation) on remote targets are predicted and displayed by the postprocessor. However, these blast effects are not presently used to predict blast enduced rupture of tanks. A simplified approach to this problem was attempted in the present study, but was found to be overly sensitive (either no tanks ruptured, or they all did).

Future versions of DERACS should include some attempt at realistically accounting for blast on the structural integrity of neighbouring tanks.

Another weakness is in the subroutines that account for the effects of explosions is the way that BLEVEs are predicted. At present a single subroutine is used to predict the occurrence of a BLEVE and this is based on a very simple comparison of lading temperature at time of puncture with published data for the superheat limit locus. Research is ongoing in this area and if additional information comes forth for predicting the probability of BLEVEs it should be incorporated into the DERACS simulation. As well the method for selecting the explosive efficiency for the case of unconfined vapor explosions is very simplistic. Efficiencies are assumed based on whether or not a BLEVE occurs and the resulting free field over pressure calculations are based on these simple assumptions of explosive efficiency. Future versions of DERACS should have more deterministic approaches to determining the explosive efficiency of the expelled vapor cloud.

8.2 Program Strengths and Potential Applications

As noted in the previous section, the present version of DERACS has many weaknesses. These weaknesses are present because of certain assumptions, many of which were necessary to limit computer run time and memory demands. However, DERACS in its present form is capable of simulating a derailment accident with some degree of realism. This in itself is significant, considering the number and complexity of the calculations involved.

The basic elements of the simulation are accounted for, and are based on cause and effect as opposed to statistics. This gives the program potential as a predictive tool.

A predictive tool could have risk analysis applications. For example, it may be desirable to study the effects of placement of tanks in certain orders in train consist. DERACS could be run to see some of the effects of tank position. By performing simulations like this, an analyst may gain considerable insight into the derailment event, realizing of course that the simulation is a simplified version of reality.

DERACS as a simulator could have significant training potential. For example, if future versions of DERACS included the effects of water spray for cooling and release containment, then response personnel could practice their tactics using the DERACS simulation. Firefighters could select water spray locations during the derailment and the simulation would account for the spray. Predicted explosions would immediately note firefighter casualties, etc. This type of simulation is of course not possible using the present version of DERACS, but could be possible in future versions.

Another application of DERACS could be for accident reconstruction. By manipulating DERACS, it could be possible to reconstruct the events in a derailment, thereby gaining additional insight from the experience. This type of reconstruction is used in the nuclear power industry to learn and document accident consequences.

Computer models of this type can also be used as forecasting tools. However, significant further development and validation is needed before DERACS could be an on site forecasting tool.

The nuclear power industry has for many years been using complex modelling tools. Additional applications for DERACS can likely be found by performing a literature survey of the use of computer models in the nuclear industry.

9. CONCLUSIONS AND RECOMMENDATIONS

9.1 Conclusions

Based on the results of the present study, the following conclusions have been made:

i) A prototype computer program has been assembled that can simulate the major events in a derailment accident.

ii) The concept of assembling various derailment event models into a single overall derailment simulation has been shown to be feasible.

iii) Although the simulation does run on an IBM AT compatible computer in a reasonable length of time, it appears that future versions of the program should take advantage of the more powerful machines available (e.g. 20 MHz 80 386 based machines or graphics work stations such as the Domain Apollo).

iv) The DERACS prototype has significant limitations due to the scope of the project and therefore should not be released without further verification and validation work being carried out.

v) The modular nature of the simulation approach allows independent improvement on a module by module basis and provides a flexible base for the addition of future modules.

9.2 Recommendations

Based on the findings of the present work, the following recommendations are made:

i) Potential applications for the DERACS program should be evaluated. This could be done by looking at how the nuclear power and aircraft industries have used computer simulations to improve safety in their industries.

ii) The results of the application study should then dictate the areas in DERACS that should be further developed.

iii) The present DERACS program should not be released to the public without proper verification, validation, etc.

iv) The DERACS program should be developed for application on more powerful computers so that some of the limiting factors (number of tanks, number of commodities, etc.) can be eliminated.

v) The DERACS program should be enhanced to a more realistic and useful level so that the simulation concept can be more effectively demonstrated to less technical potential users.

10. REFERENCES

1) Belore, R. Buist, I., A Computer Model for Predicting Leak Rate of Chemicals from Damaged Storage and Transportation Tanks, Report EE-75, Environmental Emergencies Technology Division, Environment Canada.

2) Roberts, A.F., The Effect of Conditions Prior to Loss of Containment on Fireball Behaviour, I. Chem E Symposium, Series No. 71, Pergamon Press, Oxford, 1982.

3) Birk, A.M., The Development and Validation of a Mathematical Model of a Rail Tank Car Engulfed in Fire, PhD Thesis, Queens University, Kingston, Ontario, Canada, June 1983.

4) Reed, R.C., Possible Mechanism for Pressurized Liquid Tank Explosions or BLEVEs, Science Vol. 203, March 1979.

5) Report of the Mississauga Railway Accident Inquiry, Hon. Mr. Justice S.G.M. Grange, Supreme Court of Ontario, December 1980.

APPENDIX A

INTERIM REPORT

INTERIM REPORT

LITERATURE SEARCH AND PRELIMINARY DESIGN OF A DERAILMENT ACCIDENT COMPUTER SIMULATION

Prepared For:

TRANSPORT OF DANGEROUS GOODS, AND THE
TRANSPORTATION DEVELOPMENT CENTRE
Transport Canada

Prepared By:

DAVIS ENGINEERING LIMITED
CONSULTING ENGINEERS
Ottawa, Canada

And:

A.M. BIRK ENGINEERING
Kingston, Canada

DAVIS File No.: 86-26

OCTOBER 1987

TABLE OF CONTENTS

PAGE

1.0 INTRODUCTION 1
  1.1 Background 1
  1.2 Objectives and Scope 2
  1.3 Summary 2

2.0 LITERATURE SEARCH 7
  2.1 Derailment Mechanics 8
  2.2 Derailment Impact Induced Ruptures 9
  2.3 Spill/Release Analysis 13
  2.4 Fire Analysis 14
  2.5 Thermal Rupture 18
  2.6 BLEVE/Explosion Analysis 22
  2.7 Plume Dispersion 25
  2.8 Other Resources 30
  2.9 Recommendations 31

3.0 SIMULATION SOFTWARE DESIGN 34
  3.1 Overview of Simulation Approach 38
  3.2 Input Preprocessor 43
  3.3 Simulation Software Development 55
  3.4 Postprocessor 63

4.0 CONCLUSIONS AND RECOMMENDATIONS 73

REFERENCES 74

LIST OF TABLES

PAGE

Table 1.3-1 Recommended Model to be Used in Simulation Software Development 3

Table 2.1-1 Derailment Mechanics Literature Search Summary 9

Table 2.2-1 Derailment Impact Induced Ruptures Literature Search Summary 12

Table 2.3-1 Spill/Release Analysis Literature Search Summary ... 15

Table 2.4-1 Fire Analysis Literature Search Summary 16

Table 2.5-1 Thermal Rupture Literature Search Summary 19

Table 2.6-1 BLEVE/Explosion Analysis Literature Search Summary 24

Table 2.7-1 Plume Dispersion Literature Search Summary 27

Table 3.0-1 Recommended Models to be Used in Simulation Software Development 37

LIST OF FIGURES

Figure 3.1-1 Overall Program Logic 39

Figure 3.2-1 Preprocessor Questioning Logic: Input Type 43

Figure 3.2-2 Preprocessor Questioning Logic: (Input Type 3) (Input for Full Simulation) 45

Figure 3.2-3 Preprocessor Questioning Logic: (Input Type 4) (Input for Simulation Where Derailment Mechanics Simulation is Bypassed) 49

Figure 3.4-1 Sample of Proposed Text Summary for Simulation 64

Figure 3.4-2 Accident View at Detail Level of Scale (Not to Scale) 66

Figure 3.4-3 Graphics Key Page 67

LIST OF FIGURES (continued)

PAGE

Figure 3.4-4 Screen Image Showing Grid for World and Local Levels of Scale 70

Figure 3.4-5 Accident View at Local Level (Not to Scale) 71

Figure 3.4-6 Accident View at World Level (Not to Scale) 72

1.0 INTRODUCTION

This interim report was prepared by Davis Engineering Limited in cooperation with its subcontractor, A.M. Birk Engineering. The work was conducted under DSS contract number 48SZ.T8200-6-6500/01-SZ. The project involves the development of a Derailment Accident Computer Model.

This report marks the first milestone in the project and is intended to summarize the findings of a literature search and the results of a preliminary design of the computer simulation software.

1.1 BACKGROUND

Over the last two decades, there has been a considerable amount of work carried out to develop modelling or analysis techniques to quantify the consequences of rail or highway accidents involving hazardous commodities. These developments have been in many areas including accident mechanics (both from vehicle motion and vehicle fire impingement standpoints), release rates and dispersion of dangerous goods and the prediction of fire and explosion effects on neighbouring targets. However, in most cases these models or techniques stood alone with little interaction with the mechanisms that occurred in parallel or series with them.

The present research project involves the development of a prototype derailment accident simulator that takes advantage of the various modelling studies that have preceded it. The present project involves the search and identification of models that can be integrated to give a computer simulation of an entire derailment accident. As can be imagined, this is a major task and therefore this first attempt has been defined in terms of developing a prototype simulation program to demonstrate the possibilities of such a computational tool.

The overall project objectives and plan was presented in a proposal from Davis Engineering Limited entitled, 'The Development of a Derailment Accident Computer Model', presented in June of 1986 to Transport Canada. The present report is a summary of the results of the literature search and the preliminary design of the overall computer program.

1.2 OBJECTIVES AND SCOPE

The objective of the overall project is to develop a prototype derailment accident computer model capable of simulating an entire derailment accident from start to finish. This will include the simulation of the derailment mechanics to predict vehicle placement, the prediction of impact and fire induced ruptures, the prediction of subsequent releases and the dispersion of those release in the atmosphere.

The objective of the tasks performed to date was to search for appropriate models in the open literature and to identify those that could be used in the derailment computer simulation. The next task was then to design, in a preliminary way, how these programs would work together, and how the various program inputs and outputs would be handled. From this position on it will be possible to carry out the development of the overall program.

1.3 SUMMARY

A literature search was conducted to identify analysis methods and existing computer models in the following areas which were identified as being required for the overall simulation.

- derailment mechanics
- impact rupture
- spill and releases
- fire effects
- thermal ruptures
- BLEVES/Explosions
- plume and puff dispersion.

In each of these areas, references were found and reviewed for applicability for the present project. The review was based on the following criteria:

- scope
- features appropriate for the application
- computational complexity and the computer power required
- maturity in terms of documentation, validation, etc.
- availability of code or equations
- development requirements.

Based on the above criteria, models and code have been recommended. Table 1.3-1 presents a summary of the models recommended.

SUBMODEL	SOURCE	STATUS
Derailment Mechanics	- Yang and Manos in house	- to be further developed to allow for uncoupling to facilitate possibility of side on impacts
Impact Rupture	- Yin - Phillips - to be coded	- to be further developed to include side impact ruptures - impact model in the form of equations to be coded
Spill/ Release	- Fingas - "LEAKRATE" in house	- minor modifications for integration
Fire	- Crocker - "HOTPLT" in house - Birk	- Crocker analysis to be coded - minor modifications for integration - some fire effects to be handled by tank thermal model
Thermal Rupture	- Birk - "TANKCAR" in house	- Modifications to simplify for improved speed
BLEVE/ Explosion	- Reid - Leushcen and Birk in house	- Reid analysis to be coded - minor modifications for integration with other models
Plume Dispersion	- Morgan - "SLAB" to be procured - Peterson in house	- Minor modifications for integration - Puff model to be integrated other models

TABLE 1.3-1 RECOMMENDED MODELS TO BE USED IN SIMULATION SOFTWARE DEVELOPMENT

The above identified software that is not presently in house will be obtained. Based on the literature search, procurement of the above software will be within the budget constraints. It should be noted that all of the authors will be notified of the project intent to use their software.

There are no problems anticipated with obtaining the software. During a later task (software review) it will be determined whether the development requirements are in line with expectations. With the various models identified, it was then possible to prepare a basic design for the overall simulation.

The basic simulation approach remains the same as outlined in the original proposal to carry out this problem. The simulation will involve the calling of the main calculation subroutines in the following order (note that there are now additional submodels listed which will stem from those already discussed);

1) vehicle mechanics
2) impact rupture
3) BLEVE/explosion
4) explosion blast
5) explosion thermal
6) spill size
7) gas release rate
8) fire effects
9) plume track
10) puff track
11) fire impingement
12) blast rupture
13) thermal rupture

The simulation will involve calling the above subroutines for each car in the derailment. Markers will be set on each car to identify critical conditions such as whether it is exposed to fire, whether it has ruptured, whether it is leaking gas or liquid, etc. Each subroutine will only be called if a marker states that the car under consideration require it. For those subroutines that are called to analyze spills, fires, etc., they will be called for each item separately. Items that will be treated individually then are:

1) cars
2) spills
3) releases
4) plumes
5) puffs
6) explosions
7) fires

Performance of the simulation will involve data input using a preprocessor, performance of the simulation calculations using the above referenced models and preparation of the simulation output using the postprocessor.

The input preprocessor will be menu driven. The total number of inputs for the present simulation will be kept to a minimum for the prototype software but the software will be engineered so that future development activities to enhance the capabilities are not restricted.

The postprocessor will present the results of the simulation in graphic and text form and will include accident details such as:

- vehicle placement
- surroundings
- compass, wind direction and speed in vector form
- ruptured vehicles (impact induced)
- ruptured vehicles (thermally induced)
- spills
- pool and torch fires
- vehicles exposed to fires
- plume tracks
- puff tracks
- explosion thermal effect zones
- explosion shock effect zones.

Based on the results of the literature search and the preliminary software design, it was concluded that the various submodels required to develop the accident model are available either in coded or uncoded form. The development required to put these various submodels into the desired format and condition is believed to be within the technical scope of this project. The only submodel identified as requiring significant modification or development is the derailment mechanics submodel. It was concluded that this submodel should be modified to account for vehicle uncoupling so that tank side coupler impacts can be accounted for. Also, the model should be developed to allow for tank roll so that burning flare impingement on other cars can be accounted for. Although this task is technically feasible within this project, it may require a small adjustment to the project budget.

Based on the above conclusions, it is recommended that the project continue as described in the original project proposal. It is recommended that the derailment mechanics submodel modifications (to allow for uncoupling and roll) be investigated and budgetary and schedule impacts be estimated and presented to the scientific authorities for consideration.

2.0 LITERATURE SEARCH

A literature search has been performed to identify and locate existing computer codes which would be used as submodels in the Derailment Accident Computer Simulation Model "DERACS". In addition to locating software which covered the technical scope and features required by the submodel, the literature search focussed on obtaining computer codes which had the following characteristics:

1) Availability for use in a project such as this: This requires access to the source code listing in order to have the capability of changing, or rearranging, the code to make it compatible with the hardware to be used and the software of the other submodels.

2) Not beyond the hardware capabilities for which DERACS is intended: Submodels which require excessive memory, or advanced computing facilities such as supercomputers were not suitable.

3) Validated against experimental data: This is necessary to verify the underlying theory used in the predictions.

4) Cost efficiency: The submodel acquisition cost must not be beyond the scope of the project budget.

Supporting literature, which was necessary for better technical understanding of the underlying physical processes and modelling techniques of the submodels, was also obtained.

If a computer code was not found for a particular submodel, equations and correlations from the supporting literature were identified as the information required to create coding for the submodel. For example, no specific computer code was located for actual puncture predictions in tank car walls (see Subsection 2.2). It was ascertained that this is a state-of-the-art field of research, and that codes are now being developed in U.S. DOT sponsored work (2.2-1, 2.2-2). Hence, existing equations for this were identified from work in the early 1970s by Phillips (2.2-9).

This Section presents descriptions of the various documents which were found during the literature search.

It is subdivided into subsections describing the literature for each of the submodels. Each subsection is supported by a summary table listing salient features of the computer codes and the modelling described in the papers. The documents are listed in the Reference Section and indexed according to the subsection in which they appeared, e.g. a tank puncture paper will have a designation 2.2-x. Some computer codes may be useful for more than one submodel. These are identified as cross-references in the tables, and are listed in the submodel reference section to which they are more applicable.

Recommendations on which submodel code to use are listed in the tables and reasons are delineated in the final subsection.

Finally, this literature survey is not comprehensive, nor exhaustive. However, the literature which was found is representative of the work which has been performed in those particular fields. The objective of the search was to identify a range of codes and modelling, if possible, in order to make a fair comparison between models, and to choose the code which was best suited to this application. This has been accomplished to a reasonable degree given the resource constraints, and breadth and range of the information which required searching. During the course of the computer modelling phase, other submodels may come to light, and will be given due consideration.

2.1 DERAILMENT MECHANICS

A literature survey in this area has revealed little in the way of available software for calculation of the motion of coupled railway cars when a derailment has occurred. In fact, only a single such analysis, with an accompanying software package, appears to be in existence (Yang et al. [2.1-1, -2].

The results of the literature search are summarized in Table 2.1-1. The Yang model assumes the train is derailing on a flat 2-D surface (i.e. a three degree of freedom model x, y, yaw). When a car derails it is assumed that there is suddenly no track under the car to guide it. Initial conditions determine which side of the track the car derails on. During the derailment there are only two forces acting on the car, friction forces between the wheels and the ground (including the application of brakes) and coupler forces. Since rotation of the

TABLE 2.1-1 DERAILMENT MECHANICS LITERATURE SEARCH SUMMARY

REF. NO.	AUTHOR(S)	PROGRAM NAME	SCOPE	FEATURES	AVAILABILITY	COMPUTER REQUIREMENTS	VALIDATION	FORMAT	COST	RATING
2.1-1	Yang et al	No name	-model to predict mechanics of train derailment -includes coupler forces, braking action and ground friction -2-D -3 degree of freedom (x,y, yaw)	-cars are asummed to be coupled during derailment -cars modelled as sticks with couplers acting as pin joints	-Public Domain - code is in-house at DAVIS	-FORTRAN compiler -can be used on PC with modification	-limited data available for validation	- equations in report - code included	- NIL	-recommended with modific-ations to allow for uncoupling and roll
2.1-2	Yang et al	No name	-model to predict mechanics of train derailment -includes coupler forces, braking action and ground friction	-cars are asummed to be coupled during -cars modelled as as sticks with couplers acting as pin joints	-Public Domain - code is in-house at DAVIS	-FORTRAN compiler -can be used on PC with modification	-limited data available for validation	- equations in report - code included	- NIL	-recommended with modific-ations to allow for uncoupling and roll
2.1-3	Phillips et al (1972)	No Code	-data on shelf coupler	-coupler failures and examples of uncoupling	-Public Domain	-N/A	-N/A	-discussion of data	-N/A	-of general interest
2.1-4	Ward et al (1974)	No Code	-prediction of coupler dynamic forces	-includes friction damping effects and elasticity	-Public Domain	-simple equation	-validated with with measured data	-equations	-NIL	-recommended

vehicles is not accounted for, moments are not considered. The kinetic energy of the car is dissipated by the friction forces.

In the derailment, the cars are represented as sticks with pin joints as couplers. That is, the cars have no width dimension and therefore can take up rather unrealistic positions. It is assumed in the analysis that the cars remain coupled. The program predicts coupler forces based on the assumption that the system acts as rigid links.

The analysis by Yang et al. is very well documented and quite complete, given the time frame in which it was developed. Many parts of the analysis will be useful in this study but there are significant factors which have not been taken into account and which must be included in any predictions of the results of a derailment. The major problems with the existing analysis are to:

1. The railcars are assumed to remain coupled after the incident.

   This assumption is more appropriate today, after the introduction of shelf couplers, than it was in 1972 when it was first made but is still not completely justified. Phillips and Role [2.1-3] point out that shelf couplers help to prevent override in on-line and switchyard accidents and tend to keep cars mated longer during derailments. The key word here is "longer". There can be no guarantee that the cars will actually remain coupled after a derailment. It is safe to assume that the tendency of couplers to slip apart due to vertical motions will be severely curtailed, but there is still a likelihood that couplers will be broken when subjected to the forces acting between the cars during the derailment. The majority of punctures in tank cars continue to be caused by impact with the free coupler of a neighbouring car. This must be taken into account in the model.

2. The entire derailment is assumed to occur in two dimensions.

   This assumption is clearly incorrect since many derailments will occur in areas where the track is elevated above the surrounding countryside. Railcars are likely to experience excursions outside of the horizontal plane of the rails, characterized by large angular motions in roll and pitch as well as vertical motions under the influence of gravity. None of these effects are included in the existing model.

The model to be used in the current study must, in some ways, account for these important features of a derailment. Ideally, each railcar should be given six degrees-of-freedom so that the total three-dimensional motion can be described and the final position of the car can be predicted accurately. Practically, this approach cannot be considered for a mathematical model which must run within a limited time on a micro-computer system.

A compromise solution which clearly will not provide totally accurate results but will produce predictions of the most important angular variable is to include a roll degree-of-freedom in the derailment mechanics model so that the cars which are overturned after the accident can be identified. These cars are of utmost importance to the subsequent predictions of spills and fires.

2.2 DERAILMENT IMPACT INDUCED RUPTURES

Ideally, it was envisioned that the Derailment Impact Induced Ruptures submodel would use the output from the Derailment Kinematics submodel (velocities, forces) to predict ruptures in the tank car shell walls. The ruptures would be most likely due to the impact of a coupler into the head shield or into the tank side. The rupture consequences would also be predicted, i.e. whether the rupture would lead to an explosion or BLEVE, or whether loss of the lading through leakage would occur.

The results of the literature search for the modelling of punctures due to impacts is summarized in Table 2.2-1.

The literature survey showed that this is a very current field of research. Orth (2.2-1), who is currently researching puncture dynamics on chlorine bearing tank cars for the U.S. Federal Railroad Administration, was not aware of any recent mathematical modelling. Ehrinbeck (2.2-2) from the Transportation Systems Centre, in

TABLE 2.2-1 DERAILMENT IMPACT INDUCED RUPTURES LITERATURE SEARCH SUMMARY

REF. NO.	AUTHOR(S)	PROGRAM NAME	SCOPE	FEATURES	AVAILABILITY	COMPUTER REQUIREMENTS	VALIDATION	FORMAT	COST	RATING
2.2-3	S.K. Yin (1980)	-Program incl. -no name	-model developed for study of longitudinal-vertical train action -simulates mechanisms which might lead to coupler override and tank head puncture -classification yard accidents only -up to 8 degrees of freedom -lading dynamics considered	-head puncture forces -elastic deformations -car mass concentrated at one point -symmetric impact: pitch, roll, horizontal and vertical motions	-Public domain		-verified versus	-FORTRAN source code in paper experimental results	-	Recommended
	Supporting Documentation									
2.2-6	K.H. Hohenemser (1975)									
2.2-5	D.A. Peters (1977)									
2.2-4	D.A. Peters (1980)									
.2-7	J.B. Raidt (1977)	Detailed Vertical Train Stability Model	-model developed for study of longitudinal-vertical train action -work which preceded Yin	-vertical motions only -2 dimensional model -three degrees of freedom; -longitudinal lading dynamics	-Public domain -source code available from AAR, Chicago		-limited verification	-FORTRAN	-	-Not Recommended (outdated)
	Supporting Documentation									
.2-8	J.B. Raidt et al. (1975)									
.2-9	E.A. Phillips (1972)	No Code	-model for tank head punctures velocity calculations	-threshold puncture	-Public domain		-limited verification	-Equations in report	-	Recommended

Cambridge MA, confirmed that there has been no new work on simulation since 1975. Current research that FRA and TSC are conducting will lead to new modelling in the future, perhaps in as early as one year.

Yin's work (2.2-3) was a final report to the FRA on a vertical train action model and puncture analysis in classification yard accidents. The report included a code written in FORTRAN. This work follows the progress reports on the same project written by Peters (2.2-4, 2.2-5) and Hohenemser (2.2-6) and is the work referred to by Orth and Ehrinbeck. The code was developed to simulate mechanisms which might lead to coupler override and tank head rupture by calculating impact and coupler forces. The model considers lading dynamics of the railcars and allows eight degrees of freedom: pitching, rolling, horizontal and vertical motions.

That the model was developed for classification yard accidents only is a major shortcoming of this code. No information is given for the effectiveness of head shields, for example, in mainline accidents. In those cases, the energies involved are so high that shields are ineffective and rupture is inevitable. However, it is believed that the code can still be used to predict coupler override in these cases. The code has been experimentally verified in actual full scale classification yard accidents, with good results.

The model by Raidt (2.2-7, 2.2-8) is the precursor to the work by Yin and co-workers. The code is limited by a two dimensional analysis, and only three degrees of freedom: longitudinal, vertical and pitch rotation.

The work by Phillips (2.2-9) could be used in conjunction with the code by Yin to predict punctures in tank car walls. Phillips gives, in algebraic form, equations for predicting threshold puncture velocities. The input to these equations would be the forces and velocities calculated by Yin.

2.3 SPILL/RELEASE ANALYSIS

The objectives of the Spill/Release Analysis submodel are to identify the commodities which were spilled, their properties, and to estimate the types and rates of releases. The first objective is a bookkeeping function, which identifies the puncture location, i.e. which tank

TABLE 2.3-1 SPILL/RELEASE ANALYSIS LITERATURE SEARCH SUMMARY

F.	AUTHOR(S)	PROGRAM NAME	SCOPE	FEATURES	AVAILABILITY	COMPUTER REQUIREMENTS	VALIDATION	FORMAT	COST	RATING
3-1	W.G. England et al (1985)	POOL, DROP	-leaks and evaporation of aircraft fuel from holding tanks and pressurized fuel transport lines -Cross-reference with Plume Dispersion submodel	-liquid pool spill model -time andependent -heat and mass transfer effects -differential evaporation rate equations	-Public Domain		-No Validation	-differential equations -algorithms -no source code		Not Recommended (no source code, no validation)
-4	M. Fingas et al (1986)	LEAKRATE	-leaks and atmospheric evaporation of hazardous substances from tank cars -Incorporates a leak model from ruptures in tanks and an atmospheric release model for input into dispersion codes	-experimentally derived evapoarative cooling model -effects of solar radiation, wind speed, ambient temperature, heat transfer from ground -direct (algebraic) evapoaration model -tank release model position, hole size, hole shape, and insulation shape, and insulation present -vapour, 2-phase and liquid flow models -direct (algebraic)	-Public Domain, available from Environment Canada	IBM PC	-evaporation model correlated with experimental results -tank leak model not validated as a whole; individual components tested versus versus experimental data	-Source code on 5¼" floppy disk	In House	Recommended
2 3	W. Stiver et al (1984) **Supporting Documentation** D. MacKay et al (1980)	No Code	-evaporation rate of spills of volatile liquids -developed for crude oils -used as a starting point for Fingas et al (1986)	-physical properties; temperature, wind speed, and spill dimension effects -correlation constants required -time dependent -atmospheric turbulence	-Public Domain		-validated versus small scale tunnel experiments, -fair accuracy	-algebraic equations given in paper -correlation constants given		Not Recommended

TABLE 2.3-1 (continued) SPILL/RELEASE ANALYSIS LITERATURE SEARCH SUMMARY

F. .	AUTHOR(S)	PROGRAM NAME	SCOPE	FEATURES	AVAILABILITY	COMPUTER REQUIREMENTS	VALIDATION	FORMAT	COST	RATING
3-5	M.T. Fleischer (1980)	SPILLS	-evaporation of chemical spills on land -instantaneously formed pools of liquids and liquified gases, continuous spills (leaks from tank cars), and stacks -cross-references with plume dispersion submodel	-36 chemicals -amunt of material emitted -requests emission rate through rupture, assumes this equal to emission rate to atmosphere -request pool size in instantaneous release	-Public Domain -Available from NTIS, Springfield VA	IBM 370 used previously	-limited validation	-tape medium -source code -FORTRAN	$500 US	Not Recommended (release rates assumed known)
-6	F.T. Dodge, et al (1980)	Venting Rate Submodel from HACS	-time history of tank conditions, and venting of gas and/or liquid from tank rupture of a marine cargo vessel	-evaporation model and and outflow rate	-Public Domain		-small scale tests		?	Not Recommended (not applicable to rail cars)
-7	P. Shaw et al (1978)	SPILL	-simultaneous spread and vaporization from spills of hazardous liquids -cryogenic liquids, e.g. LNG, LPG	-heat transfer effects -emission rate output -cryogenics or other hazardous liquids	-Public Domain -Available from SRD, UKEA, England	-mainframe computer	-verified against published -experimental -reasonable accuracy	-source code -tape medium -FORTRAN	£1000	Not Recommended (outdated model)

TABLE 2.4-1 FIRE ANALYSIS LITERATURE SEARCH SUMMARY

REF. NO.	AUTHOR(S)	PROGRAM NAME	SCOPE	FEATURES	AVAILABILITY	COMPUTER REQUIREMENTS	VALIDATION	FORMAT	COST	RATING
2.4-1 2.4-1	G. Schneyer, et al (1975) Supporting Documentation L. Schalit et al (1974)	TORCH	-torch fire simulation model -diffusion flames impinging on rail tank cars -computer code never successfully completed -steady state, axisymmetric profiles of temperature, velocity and component mass-fractions	-differential equations: solution to conservation of mass, momentum, species and turbulent kinetic energy -finite difference formulation -Spalding turbulence model -models soot and radiation transport		-mainframe computer	-code not successfully validated	-equations in paper		Not Recommended (too extensive, not successful)
2.4-4	J.L. Rand et al (1982)	HOTPLT	-calculates heat transfer from a combusing propane jet impinging on a flat plate -vapour torch flame	-1-dimensional solution -calculates heat transfer coefficients	-Public Domain		-field tests on 1/5 scale torch simulation facility -correlation coefficients for agreement of code with experimental results	-FORTRAN source listing in paper	-	Recommended
.4-5	A.M. Birk et al (1982)	Part Of Tankcar	-radiant heat trans. to long cylinder engulfed in fire	-integration of view factors -variable flame absorption	Public Domain		-as part of Birk tank thermal model	FORTRAN		- Recommended
.4-6 .4-7	M. Tunc et al (1984) M. Tunc et al (1984)	No Code	-models to predict heat transfer from pool fires and venting flares to horizontal cylinder -3 dimensional formulations: variation of fluxes along and around circumference	-integral equations -cross-wind modelled -gray gas approximation -assumptions of flame and tank temperatures	-Public Domain		-claim reasonable accuracy upon comparison with experimental -no experimental results presented	-equation (algebraic integral in paper)	-	Not Recommended

TABLE 2.4-1 (continued) FIRE ANALYSIS LITERATURE SEARCH SUMMARY

F. .	AUTHOR(S)	PROGRAM NAME	SCOPE	FEATURES	AVAILABILITY	COMPUTER REQUIREMENTS	VALIDATION	FORMAT	COST	RATING
4-8	W.P. Crocker et al (1987)	No Code	-models to predict the heat trasnfer from pool fires and fireballs -used for hazard (damage) predictions -work performed for highway trucks; applicable to any vehicle carrying flammable liquid and liquefied petroleum gases -predicts shape, size, temperature and thermal emissive power of pool fires and fireballs	-confined and unconfined pool fires -instantaneous, continuous quasi-continuous releases -burning grate, flame length, flume tilt, atmospheric attenuation growth rate -Probit dosage equations	-Public Domain		-no validation -parameter study included	-equations (algebraic, integral listed in paper	-	Recommended
-9	D.A. Lihou et al (1982)	No Code	-radiation predictions for fireballs -correlation of small scale experimental results; applicable to BLEVEs	-hazard ranges for people from BLEVEs and fireballs -correlation of hazard ranges -hot shell, cold ore method -butane and mehtane gases	-Public Domain		-validated for small scale (soap bubble)	-equations (algebraic, given in paper	-	Not Recommended
-10	A.F. Roberts (1982)	No Code	-cloud formation and fireballs after BLEVE	-size, duration, radiation of fireballs -primitive attenuation model	-Public Domain		-no Validation	-algebraic equations	-	Not Recommended
-11	J.S. Klement (1986)	No Code	-effects of thermal radiation at a particular site after an explosion of a rail tank car carrying flammable liquids	-instantaneous vapour release (Caussian puff) -fireball, pool fire and blast wave analysis	-Public Domain -thesis copyright by author		-no Validation	-algebraic equations	-	Not Recommended (site specific, not general) -method useful

Birk (2.4-5) and Tunc (2.4-6, 2.4-7) and their respective co-workers studied the effects of pool fires on tank cars.

Birk (2.4-5) developed a thermal radiation model of a long cylinder engulfed in fire. The fire heat transfer is calculated as a function of the fire and tank surface temperatures, the flame shape and the radiating characteristics of the fire. The model is based on the gray gas assumption. This fire model eventually became part of a more comprehensive tank - car thermal model (to be discussed in section 2.5) and was validated as part of that model.

Tunc (2.4-6, 2.4-7) and co-worker developed analytic integral models to predict heat transfer to a finite length cylinder from pool fires and venting flare fires. These were three dimensional models, which output fluxes along and around the circumference of the cylinder using assumptions of flame and tank temperatures. Features of this modelling include crosswind modelling and the gray gas approximation for the fire. The authors claim reasonable accuracy upon comparison with experimental results; however, no evidence of this is presented.

Whereas Birk and Tunc presented work for heat transfer to tanks engulfed in fires, Crocker and co-workers (2.4-8) studied heat transfer from pool fires and fireballs to the "outside world". The models which the authors present were developed for highway tanker truck accidents, but are applicable to any vehicles which carry flammable liquids. The models were intended to predict hazard or damage distances. The paper furnishes equations for the prediction of shape, size, temperature, and thermal emissive powers of pool fires and fireballs, as well as burning rate, flame length and tilt, growth rate, and atmospheric attenuation of radiation. The models are applicable to confined and unconfined pool fires, and for instantaneous or continuous releases. The models have not been validated; however a parametric study is presented.

Lihou and co-workers (2.4-9) and Roberts (2.4-10) present models for radiation from fireballs resulting from BLEVEs. Lihou's models were validated for small scale (soap bubble) experiments.

A recent Master's project report by Klement (2.4-11) presents an analysis of the consequences of an explosion of a rail tank car carrying propane, at an arbitrarily chosen site. Klement uses various correlations of heat transfer from fireballs and pool fires, as well as blast wave analysis. Hand calculations were used throughout the analysis.

2.5 THERMAL RUPTURE

The purpose of the Thermal Rupture submodel is to predict the conditions of a laden tank car which is exposed to a fire environment. This entails heat transfer to the tank car due to pool fire engulfment, torch impingement, or radiation from a nearby fire such as a pool fire or fireball. The Thermal Rupture submodel would use input from the Fire Analysis. The output would include pressures and temperatures inside the tank, whether the pressure relief valve was venting to the atmosphere, and time to rupture. The Thermal Rupture submodel is central in the overall accident model: its output would be used in subsequent fire analysis (flares and pool-forming leakage) if the lading is flammable; in the BLEVE and Dispersion Model in the case of a rupture; and in the Spills/Release and the Dispersion model in case of volatile non-flammable liquids.

The literature search located several programs for the Thermal Rupture submodel of varying degree of analytical rigour. The results of the search are displayed in Table 2.5-1.

Almost all tank thermal rupture research references the work by Graves (2.5-1), which describes an unnamed computer code for the calculation of temperatures and pressures inside a tank car engulfed in a fire. Included in the code is the prediction of the mass of lading remaining in the tank after pressure relief, and the location of the liquid level.

Although not totally validated this program appears to have shown the way for many of the models that have been developed since this work was conducted.

A recent thesis by Wentzell (2.5-2), as well as work by Sousa and co-workers (2.5-3) describe an unnamed computer code which evaluates the thermal response of LPG and LNG tanks exposed to pool fires. The code calculates pressures and temperatures inside the tank using time-dependent, finite difference formulations of the conservation of mass, energy and volume equations; as well

TABLE 2.5-1 THERMAL RUPTURE LITERATURE SEARCH SUMMARY

REF. NO.	AUTHOR(S)	PROGRAM NAME	SCOPE	FEATURES	AVAILABILITY	COMPUTER REQUIREMENTS	VALIDATION	FORMAT	COST	RATING
2.5-1	K.W. Graves (1973)	Program Included (No name)	-model to calculate the temperatures and pressures inside a tank car engulfed in a fire	-tank pressure -temperature of liquid and vapour -temperature of tank shell -mass of remaining liquid -location of liquid level	-Public Domain			-source code listing in paper -FORTRAN	-	Not Recommended (outdated model)
2.5-2	P.R. Wentzell (1985)	Program Included	-model evaluates thermal response of LPG and LNG exposed to a fire environment -pool fire environment	-time dependent, finite difference code using conservation of mass, energy and volume -explicit equation for pressure-time derivative -calculates pressures and temperatures -lumped analysis for lading -statified or bubble rise liquid/vapour interface -equation of state interpolated from tables -all interior and exterior heat transfer modes accounted for -pressure relief valve	-Public Domain -Thesis copyrighted by author		-validated against one fifth scale published experimental data -uncertainties in measured quantities -appears to be good equipment	-source code listing in thesis -FORTRAN	-	Recommended
2.5-3	Supporting Documentation A.C.M. Sousa et al (1985)									
2.5-4	A.M. Birk (1985)	Tank Car Thermal Computer Model (TCTCM)	-Model evaluates thermal response of rail tank car model exposed to a fire environment -pool and torch fire environment	-time dependent, finite difference code -predicts tank pressures, wall and lading -pressure relief valve -predicts tank failure time and lading properties at that time.	-In house at DAVIS	-Code runs on VAX 730 and IBM PC-AT	-validated against published -good agreement	-FORTRAN source code -5¼" disk for IBM PC	-	Recommended

TABLE 2.5-1 (continued) THERMAL RUPTURE LITERATURE SEARCH SUMMARY

REF. NO.	AUTHOR(S)	PROGRAM NAME	SCOPE	FEATURES	AVAILABILITY	COMPUTER REQUIREMENTS	VALIDATION	FORMAT	COST	RATING
2.5-5 2.5-6	M.R. Johnson et al (1984) Vols. I and II	FEAVIT	-model to calculate temperatures and pressures inside a tank car engulfed in a fire -pool fire environment	-algebraic equations -used to evaluate various conductances and different flow capacities of pressure relief valves -upright or overturned tank -burst strength modelled as a function of wall temperature -predicts time to failure and lading properties at that time	-Public Domain		-No Validation	-FORTRAN source listing in paper	-	Recommended
2.5-7 2.5-8	D.L.M. Hunt et al (1985) **Supporting Documentation** K. Moodie et al (1985)	ENGULF	-model to calculate temperatures and pressures inside a tank car completely engulfed in a pool fire (no partial engulfment)	-time dependent model -cube model of tank -liquid/vapour control volumes -interior and exterior heat transfer modes accounted for -fourth order Runge-Kutta mathematical solver	-Public Domain -available from HSE, UKAEA, England		-validated against published experimental results -appears to be good agreement	-tape medium -FORTRAN source code	£1000	Not Recommended (simplistic)
2.5-9	Beynon et al	HEATUP	-model to calculate temperatures and pressures inside a tank car completely engulfed in a pool fire (no partial engulfment)	-details unknown	-shell research limited		-validated against scale fire tests	-		-not recommended (details of model unkown)

as the equations of state (interpolated from tables). However, the code used an explicit equation for the pressure-time derivative, i.e. the pressure rise behaviour. The pressure relief valve is modelled. The code was validated against published one-fifth scale experimental data, and despite uncertainties in the measured quantities, the code appears to be in good agreement with the test results.

Birk (2.5-4) developed a code called "TANKCAR" which evaluates the thermal response of a rail tank car exposed to pool and torch fires. The code predicts the tank pressure, and shell and lading temperatures using a quasi-three-dimensional finite difference formulation. It also predicts lading venting through the pressure relief valve, erosion of insulation, tank failure time through weakening of the tank material due to heat, and lading properties at the time of failure. The code was validated against published experimental data, with good agreement.

Johnson and co-workers (2.5-5, 2.5-6) developed a code call "FEAVIT" to predict the temperatures and pressures inside a tank car engulfed in a pool fire of uniform intensity. The code was developed primarily to evaluate the effectiveness of insulation of varying thickness and conductivity, and the different flow capacities of various pressure relief valves. The code uses time-dependent algebraic correlations to calculate time-to-failure, and lading properties at that time, for upright or overturned tanks. Verification of the accuracy of the code is not provided.

Hunt (2.5-7) and Moodie (2.5-8) and their respective co-workers describe a code called "ENGULF", which calculates time-dependent temperatures and pressures inside a tank car entirely engulfed in a pool fire. The code is based on a cube model of the tank, made to give the same interior volume and wall surface area as the cylindrical tank. The system representation involves a 12-noded thermal network, whose governing equations are solved by a fourth order Runge-Kutta mathematical solver. Algebraic equations are employed for the boiling and venting of the lading. The model was verified against published experimental results, with which it appears to be in good agreement.

Beynon (2.5-9) and co-workers describe a code named "HEATUP" which is capable of predicting the time response of a tank exposed to engulfing fires. The model is a 2-D representation. The program accounts for fire radiation, tank wall (multi layered) radial conduction, and convection and radiation in the lading. Several pressure relief valve action models are available. This program has been compared with a number of fire tests and agreement appears good.

2.6 BLEVE/EXPLOSION ANALYSIS

As described by McDevitt and co-workers (2.6-1), a Boiling Liquid Evaporating Vapour Explosion (BLEVE) is a term used to describe the phenomenon created by the sudden loss of containment and depressurization to atmospheric of a liquid. The phenomenon is characterized by the creation of a blast wave due to the rapid boil-off of any liquid.

Hence, this is a physical phenomenon which can occur to any pressurized liquid which is superheated above the normal boiling temperature at atmospheric pressure. Of course, if the liquid is flammable, and there is a source of ignition nearby, a fireball will occur after the formation of a vapour cloud following the rapid boil-off.

The purpose of the BLEVE/Explosion Analysis submodel is to determine whether a BLEVE or other type of explosion will occur. If such an explosion occurs, the blast wave, and its effects, must be calculated, as well as the radiation effects from the fireball if the liquid is flammable. The submodel would use the output of the Thermal Rupture and Derailment Impact Induced Rupture submodels to determine whether loss of containment occurs. The results of the BLEVE/Explosion Analysis submodel would be used in the Fire Analysis and Plume Dispersion submodels.

Professor R.C. Reid of MIT is an acknowledged expert in the field of BLEVEs. He is unaware of any existing computer programs for the prediction of BLEVEs and their consequences (2.6-2). This is due to a general inadequate understanding of BLEVE mechanisms, and disagreement among researchers of BLEVE phenomena. Reid's work (2.6-3), however, can be used as a starting point for this submodel. He presents an algebraic/graphic means of predicting the onset of a BLEVE.

The literature search, therefore, concentrated on the consequence aspect of the analysis, and is summarized in Table 2.6-1.

Clancey (2.6-4) presents a general, simple model for the prediction of propagation of a blast wave from explosions. The model predicts overpressure, shock velocity and particle velocities far from the blast origin, i.e. neglecting near-field effects. The interactions of the shock with permanent structures is presented, along with a peak pressure/damage effects correlation. No verification of this work is presented.

MacKenzie and co-workers (2.6-5, 2.6-6) describe a code called "GASEX1". The program calculates the blast wave from an Unconfined Vapour Cloud Explosion (UCVE). A UCVE differs from a BLEVE in that the blast is generated by the combustion of flammable gas in the UVCE. The BLEVE blast wave is caused by the rapid phase change from liquid to vapour, which can occur to any liquid.

The code "GASEX1" consists of two parts: a one-dimensional hydrodynamics code for the blast analysis, and a hydrocarbon combustion model. The hydrodynamics part is a time-dependent, finite difference Lagrangian formulation, and can handle planar, cylindrical, or spherical geometry. The governing equations are the usual conservation of mass, momentum and energy equations, along with the equation of state. The combustion model calculates flame speeds. The code can be used without the combustion model, for use with a bursting container, using conventional techniques. The output of the code includes position of the flame front, amount burned, overpressures, local flame speed, and total energy release, among others, versus time. Graphical output is possible for the pressure distance relationship. No validation of the code is presented, but a parametric study is given.

A related code, "GASEX2", is described by Martin (2.6-7, 2.6-8). "GASEX2" also calculates the blast wave consequences from a UVCE, but does not include a combustion model. (The combustion model from GASEX1 is recommended by Martin). The code also predicts the effects of the interaction of the blast wave with rigid structures. The hydrodynamics coding is a time-dependent two-dimensional finite difference formulation of the conservation equations, using the Implicit Continuous-Fluid Eulerian method, combined with the Arbitrary Lagrangian-Eulerian method (ICED-ALE). The marker-and-cell (MAC) method of particle tracking is available as an option. The output is in the form of temperatures, positions, pressures and particle locations against time. Graphical output is available also. The

TABLE 2.6-1 BLEVE/EXPLOSION ANALYSIS LITERATURE SEARCH SUMMARY

REF. NO.	AUTHOR(S)	PROGRAM NAME	SCOPE	FEATURES	AVAILABILITY	COMPUTER REQUIREMENTS	VALIDATION	FORMAT	COST	RATING
2.6-3	R.C. Reid (1979)	No Code								
2.6-4	V.J. Clancey (1982)	No Code	-prediction of blast waves from explosions - remote from origin -peak overpressures, shock velocity and particle velocities	-interaction of shocks with structures -concept of TNT equivalent -peak pressure/damage correlation	-Public Domain		-No Validation except for the correlation	Algebraic equations	-	
2.6-5	J. MacKenzie et al (1983) Supporting **Documentation**	CASEX1	-calculation of blast wave from gas cloud explosion -overpressure from combustion of gas cloud -can be used without combustion model for use with a bursting container	-1-dimensional hydro-namics and combustion model for hydrocarbons -finite difference formulation -time dependent -Lagrangian formulation -planar, cylindrical or spherical geometry -conservation of mass, momentum and energy -equation of state -graphical output possible -output of time, position of flame front, amount burned, overpressure, etc. -125 mesh points -restart facility	-Public Domain -available from SRD, UCEAE, England	40 kBytes RAM 10 kByte disk file	-No Validation	-FORTRAN source listing, tape medium	£1000	Recommended
2.6-6	J. MacKenzie (1983)									
2.6-7	D. Martin (1983) Supporting **Documentation**	CASEX2	-calculation of blast from gas cloud explsion -interaction of shock wave with rigid structures -no combustion model	-2-dimensional hydro-dynamics code -ICED-ALE method -finite element formulation of conservation of mass, momentum and energy -marker-and-cell option for particle following -graphical output available	-Public Domain -available from SRD, UKAEA, England	-mainframe computer -CDL 7600 and ICL 2980 computers used previously	-Validated against other blast wave codes and shock tube experiments -showed good agreement	-FORTRAN source listing, type medium	£1000	
2.6-8	D. Martin (1983)									
2.6-11	Leuschen and Birk	BLAST	-blast loads on simple structures -blast effects	-1-D -analytical plus empirical -graphical output	-public domain -Queen's Univ.	-IBM/PC	-validated with exp. data for explosions	-FORTRAN		-recommended

code was compared with other blast wave codes, and shock tube experiments. The results of the comparisons show good agreement.

Texts by G. Kinney et al (2.6-9) and Baker et al (2.6-10) provide comprehensive analysis techniques for blast wave propagation and consequences.

Leuschen and Birk (2.6-11) developed a code to predict air blast loads on simple structures including cylinders, and rectangular parallelepipeds. The analysis only considers normally incident shockwaves. The program can consider a number of commodities and reduces the energy content of these commodities to TNT equivalency. The free field overpressure as a function of distance is then calculated. Loads on structures are estimated. Damage estimates for a range of distances is given as part of the output.

2.7 PLUME DISPERSION

The purpose of the Plume Dispersion submodel is to track the releases of hazardous gases which result from vaporization (non-flammable liquids) or combustion (flammable liquids). The Plume Dispersion submodel would ideally be a three-dimensional model to simulate the dispersion spread horizontally (ground level) downstream and vertically. It would use as input the output from the Spill/Release, Fire Analysis, and BLEVE/Explosion submodels.

Plume dispersion models can be generally grouped into five categories (2.7-15):

1) Gaussian puff/plume or top hat;

2) box models - simple uniform profiles;

3) advanced similarity or advanced box - more complex profiles;
4) layer averaged - conservation equations reduced to first order partial differential equations; and

5) full three-dimensional conservation models.

The literature search revealed many dispersion models in each of the five categories, and most of these had computer codes. The literature survey is summarized in Table 2.7-1.

Chan (2.7-1, 2.7-2 and 2.7-3), Morgan (2.7-4) and Hogan (2.7-5) and their respective co-workers have developed a code called "FEM3", which is a fully three-dimensional, time-dependent, finite element code for the dispersion of heavy gas in the atmosphere. This is likely the most sophisticated code available for plume dispersion: it incorporates variable ground surface profile, eddy diffusion tensor turbulence modelling, arbitrary wind profiles, heat transfer from the ground and gravitational spread of the vapours, among other features. However, it is also the most difficult program to implement computationally, requiring expensive and large-capacity hardware for reasonable computation time.

Morgan (2.7-6, 2.7-7, and 2.7-8) and co-workers have developed a simpler plume dispersion code which is computationally easier to implement. The code is called SLAB, and is a time-dependent, one-dimensional, quasi-three-dimensional program, which also solves the conservation of mass, momentum, energy and species equations for heavy gas dispersion in the atmosphere. It belongs in the layer averaged category, and uses the method of lines to solve the governing equations. Features include the incorporation of turbulence modelling, atmospheric stability, and ground heat transfer. However, only flat terrain is allowable, and cloud meander cannot be accounted for. The code has been validated against full scale experimental results, with good agreement.

Analyses by Tsang (2.7-9), Koch and co-workers (2.7-10) and Peterson (2.7-11) are examples of Gaussian puff/plume models. Each has the ability to calculate the ground concentration of a substance released from a source along the centreline of propagation, as well as plume rise. These models are computationally simple, requiring coding of algebraic equations.

Raj and co-workers (2.7-12, 2.7-13 and 2.7-14) have developed a code called "SAFEMODE", which calculates downwind concentrations of gases from an instantaneous release of heavy gas. It belongs in the box model category, with a uniform radial spread velocity profile. Turbulence modelling is included, via the entrainment, or gradient, approach. The code has been validated against full scale field tests, and produced reasonably accurate predictions.

TABLE 2.7-1 PLUME DISPERSION LITERATURE SEARCH SUMMARY

EF. D.	AUTHOR(S)	PROGRAM NAME	SCOPE	FEATURES	AVAILABILITY	COMPUTER REQUIREMENTS	VALIDATION	FORMAT	COST	RATING
.7-1	S.T. Chan et al (1983)	FEM3	-heavy gas dispersion in the atmosphere, e.g. LNG -gravitation spread and vapour dispersion (concentrations) -limited range of spill scenarios; requires validation for explosion releases	-3-dimensional finite element code -time dependent analysis -Galerkin 8-node parametric hexahedron elements -variable ground surface -heat transfer from ground -turbulence modelled	-From National Energy Software Centre, Argonne Nat'l. Labs, Argonne IL	-CRAY supercomputer -VAX -running time: up to 1/2 hour on CRAY; days on VAX	-validated versus versus expt'l. results, presented in graphical form	-tape medium -source code	$1841 US	-Not Recommended (computer requirements)
	Supporting Documentation									
.7-2	S.T. Chan et al (1983)									
.7-3	S.T. Chan et al (1983)									
.7-4	S.T. Chan et al (1984)									
.7-5	W.J. Hogan et al (1982)									
.7-6	D.L. Morgan et al (1983)	SLAB	-heavy gas dispersions in the atmosphere, e.g. LNG -no modelling of cloud meander -concentration, height, width, speed	-1-dimensional, quasi-3-dimensional model -turbulence modelled -method of lines to solve coupled partial differential equations -flat terrain -time dependent	-from National Energy Software Center, Argonne Nat'l. Labs, Argonne IL	-CDC 7600 -PC -running time: 3 seconds on 7600; 1 min. on PC	-validated versus experimental results, in graphical form, -good agreement	-tape medium -source code	$1103	-Recommended
	Supporting Documentation									
.7-7	D.L. Morgan et al (1983)									
.7-8	D.L. Morgan et al (1983)									
7-4	D.L. Morgan et al (1984)									
7-5	D.L. Morgan et al (1982)									
7-9	G. Tsang (1969)	No Program	-concentration of SO_2 and SF_6 released from a source	-steady state Gaussian plume model -plume rise, centreline contour -concentrations along centreline	-Public Domain		-validated versus field tests	-algebraic equations in paper	-	Not Recommended

TABLE 2.7-1 (continued) PLUME DISPERSION LITERATURE SEARCH SUMMARY

REF. NO.	AUTHOR(S)	PROGRAM NAME	SCOPE	FEATURES	AVAILABILITY	COMPUTER REQUIREMENTS	VALIDATION	FORMAT	COST	RATING
2.7-10	R.C. Koch et al (1971)	-Program (incl.) -No name	-urban air pollution model for diffusion from multiple sources -ground level pollution concentrations (SO_2)	-Gaussian plume model -multiple point sources -area source capability -plume rise -short term and long term integrations	-Public Domain		-validation with with selected urban pollution -short term; large deviation from measured -long term; good agreement	-FORTRAN listings in paper	-	Not Recommended (dated, inappropriate)
.7-11	W.B. Petersen (1982)	-Program (incl.) -No Name	-ground level concentrations from an instantaneously release puff	-Gaussian puff model	-Public Domain			-FORTRAN listed in paper	-	Not Recommended (too simplistic)
.7-12	Technology and Management Systems Inc. (TMSI)	SAFEMODE	-instantaneous release model -downwind concentrations from release of toxic cloud of heavy gas	-"box" model; uniform radial spread velocity profile -turbulence modeling included: entrainment or gradient approach -menu driven, user friendly	-available for purchase from TMSI	-IBM PC: 512K 360K floppy disk drive -IBM EGA monitor -DOS 3D	-validated versus field tests -reasonably accurate predictions	-floppy disk medium	$10,000	Not Recommended (no source code; cost)
	Supporting Documentation									
.7-13 .7-14	P.K. Raj (1986) P.K. Raj (1985)									
.7-15	E. Alp (1985)	COBRA	-estimates hazard zones -unconfined spills on water; later versions to handle land spills -ground level concentrations	-advanced similarity model -power law wind velocity model -transient gas releases	-available on contract basis only		-validated versus published experimental data -satisfactory results, general under predictions	-FORTRAN code on tape	-	Not Recommended (contract basis, no source code)

TABLE 2.7-1 (continued) PLUME DISPERSION LITERATURE SEARCH SUMMARY

REF. NO.	AUTHOR(S)	PROGRAM NAME	SCOPE	FEATURES	AVAILABILITY	COMPUTER REQUIREMENTS	VALIDATION	FORMAT	COST	RATING
2.7-16	T.L. Devean (1985)	ALOHA	-plume concentration calculations	-Gaussian statistics model -plot of plume contours -270 chemicals -ground roughness included -street map input possible -menu driven, user friendly	-Public Domain	-Macintosh computer		-3½" disk	-	Not Recommended (no souce code)
2.3.5	N.T. Fleischer (1980)	SPILLS	-spill/evaporation model which contains a plume dispersion model -plumes from instantenously formed pools of liquid and continuously leaking spills -concentrations as function of time and distance downwind -cross-references with Spills/Release submodel	-3-dimensional Gaussian puff dispersion model -predicts evaluation zones -time-dependent calculations -properties of 36 chemicals -constant concentration contour plots	-from NTIS, Springfield, Va.	-previously run on IBM 370 computer	-models validated validated versus experimental results -conservative predictions	-tape medium -source code	$500 US	Recommended (cross-reference)
.3-1	W.G. England (1985)	DISCO	-spills/release and dispersion models for aircraft fuel spill -cross-references with Spills/Release submodel -conservation equations for mass, momentum, energy and concentration -no pressure dependency field -eddy diffusivity turbulence model	-steady state dispersion model -quasi 3-dimensional model-finite difference	-Public Domain			-algorithm given in paper -no source code	-	Not Recommended (no source code)

Alp (2.7-15) describes a code called "COBRA", which estimates hazard zones, due to the dispersion from unconfined spills of heavy gases on water. This is an advanced similarity model, similar to the box model approach, except with power law wind velocity profiles. The validations to date showed a general underprediction trend. In addition, the code is not appropriate for this work, since it only models spills on water at present.

DeVann (2.7-16) describes a code called "ALOHA", developed to predict plume concentrations using a Gaussian statistics model. The code is extremely user-friendly, and produces plots of plume iso-concentration contours for a large number of gases.

As part of the "SPILLS" codes from the Spill/Release Analysis submodel, Fleischer (2.3-5) describes a time-dependent, three-dimensional Gaussian puff dispersion model to predict hazard zones resulting from spills. It calculates concentrations as functions of time and distance downwind from instantaneously formed pools, or continuously leaking spills. This model produces conservative predictions when compared with experimental results.

England (2.3-1) and co-workers describe a plume dispersion model as part of their Spills/Release Analysis code. The dispersion part of the code is called "DISCO", and is a steady state, quasi-three-dimensional finite difference model, in which the pressure dependancy has been replaced with stream functions. No validation for the code is offered by the author.

An often quoted text for plume dispersion work is Pasquill (2.7-17), and should be used as a background information source.

2.8 OTHER RESOURCES

Several other general sources from the literature may prove to be useful in synthesizing the overall model.

Glickman (2.8-1) and Napier (2.8-2) provide general methodologies in producing a risk assessment procedure in the transportation industry. Colonna (2.8-3) reviews the submodels in HACS. Dabberdt (2.8-4) and Ooms and Tennekes (2.8-5) have compiled collections of papers on the atmospheric dispersion of heavy gases.

2.9 RECOMMENDATIONS

As described in previous pages, the derailment simulation will require sub models to represent the following:

1) Derailment Mechanics
2) Derailment Impact Induced Rupture Analysis
3) Spill/Release Analysis
4) Fire Analysis
5) Thermal Rupture Analysis
6) BLEVE/Explosion Analysis
7) Plume Dispersion Analysis

The literature search presented in the previous section identified models for each of the above items. The following sections contain recommendations as to which of the identified models will be used in the development of the overall derailment simulation program.

Derailment Kinematics

The computer code by Young (2.1-1, 2.1-2) is the only derailment mechanics code identified in the literature search. However, this code has the failing that uncoupling and roll of cars is not allowed for. It is therefore recommended that this code be modified (or a similar code be developed) such that uncoupling and roll is accounted for. This will allow for a much more realistic simulation of impact induced ruptures.

Derailment Impact Induced Ruptures

As stated in section 2.2 this is a very active and current area of research.

The unnamed program by Yin (2.2-3) and others (2.2-4, 2.2-5, 2.2-6) is recommended for the Derailment Impact Induced Rupture submodel. It is the most up-to-date train action, decoupling prediction model. The code is available in the public domain and the results have been successfully verified. The correlations by Phillips (2.2-9) should be programmed for use in conjunction with Yin's code. It should be noted, however, that the above models are for coupler on end impacts. With uncoupling accounted for in the derailment kinematics model it will be possible to have coupler on side impacts. Therefore it will be necessary to formulate a simple model for side impact ruptures. This will be done but of course will be an area requiring future validation. The side impact model will be a simple model based on impact geometry and areas and the threshold puncture velocities published by Phillips (2.2-9).

Spill Release Analysis

The code "LEAKRATE" by Fingas and co-workers (2.3-4) is recommended for the Spill/Release submodel. It is the most comprehensive model found in its class, incorporating up-to-date modelling techniques for both evaporation and leakage prediction. It has been validated, and is available to the public.

Fire Analysis

The Fire Analysis submodel will require three aspects to it: a torch fire impinging on the tank car, analysis of heat transfer to a tank car engulfed in a fire, and heat transfer to a tank car or other remote target not engulfed in flames (fireballs, nearby pool fires). These requirements can be fulfilled by "HOTPLT" (2.4-4), Birk's (2.4-5) analysis and the work by Crocker (2.4-8). The analysis from the latter will have to be coded by DAVIS.

Thermal Rupture Analysis

Although there have been several successful attempts at Thermal Rupture analysis by several authors, Birk's work (2.5-4) is the most comprehensive and complete, and is recommended here for the Thermal Rupture submodel. It has the further advantage of having been developed, debugged and validated at DAVIS.

BLEVE/Explosion Analysis

The BLEVE/Explosion submodel will have two aspects to it - a prediction of the onset of a BLEVE, and BLEVE or explosion consequences. Reid's work (2.6-3) is widely quoted as a standard BLEVE prediction technique, and will require programming for this application. The work of Leuschen and Birk (2.6-4) will be used to predict the explosion consequences. This will be done in the form of predicted loads on simple surrounding structures and by showing damage radii. These submodels could be supplemented by the texts by Kinney (2.6-9) and Baker (2.6-10).

Plume Dispersion

The code "SLAB" (2.7-4, 2.7-5, 2.7-6, 2.7-7, 2.7-8) is recommended for the heavy gas Plume Dispersion submodel. Although not as extensive as FEM3, (2.7-1, 2.7-2, 2.7-3, 2.7-4, 2.7-5), "SLAB" is only slightly less accurate, is far easier to execute computationally, and originates from the same research environment. It is available in the public domain, for a nominal processing fee. The other codes which were located were either outdated, or not available for the type of use required in this project.

For neutrally buoyant gas dispersion, it is recommended that the puff model of Peterson (2.7-11) be used. This model can be used to model the dispersion of instantaneous releases such as explosions and also continuous releases by using a continuous series of puffs.

2.9 RECOMMENDATIONS

As described in previous pages, the derailment simulation will require sub models to represent the following:

1) Derailment Mechanics
2) Derailment Impact Induced Rupture Analysis
3) Spill/Release Analysis
4) Fire Analysis
5) Thermal Rupture Analysis
6) BLEVE/Explosion Analysis
7) Plume Dispersion Analysis

The literature search presented in the previous section identified models for each of the above items. The following sections contain recommendations as to which of the identified models will be used in the development of the overall derailment simulation program.

Derailment Kinematics

The computer code by Young (2.1-1, 2.1-2) is the only derailment mechanics code identified in the literature search. However, this code has the failing that uncoupling and roll of cars is not allowed for. It is therefore recommended that this code be modified (or a similar code be developed) such that uncoupling and roll is accounted for. This will allow for a much more realistic simulation of impact induced ruptures.

Derailment Impact Induced Ruptures

As stated in section 2.2 this is a very active and current area of research.

The unnamed program by Yin (2.2-3) and others (2.2-4, 2.2-5, 2.2-6) is recommended for the Derailment Impact Induced Rupture submodel. It is the most up-to-date train action, decoupling prediction model. The code is available in the public domain and the results have been successfully verified. The correlations by Phillips (2.2-9) should be programmed for use in conjunction with Yin's code. It should be noted, however, that the above models are for coupler on end impacts. With uncoupling accounted for in the derailment kinematics model it will be possible to have coupler on side impacts. Therefore it will be necessary to formulate a simple model for side impact ruptures. This will be done but of course will be an area requiring future validation. The side impact model will be a simple model based on impact geometry and areas and the threshold puncture velocities published by Phillips (2.2-9).

Spill Release Analysis

The code "LEAKRATE" by Fingas and co-workers (2.3-4) is recommended for the Spill/Release submodel. It is the most comprehensive model found in its class, incorporating up-to-date modelling techniques for both evaporation and leakage prediction. It has been validated, and is available to the public.

Fire Analysis

The Fire Analysis submodel will require three aspects to it: a torch fire impinging on the tank car, analysis of heat transfer to a tank car engulfed in a fire, and heat transfer to a tank car or other remote target not engulfed in flames (fireballs, nearby pool fires). These requirements can be fulfilled by "HOTPLT" (2.4-4), Birk's (2.4-5) analysis and the work by Crocker (2.4-8). The analysis from the latter will have to be coded by DAVIS.

Thermal Rupture Analysis

Although there have been several successful attempts at Thermal Rupture analysis by several authors, Birk's work (2.5-4) is the most comprehensive and complete, and is recommended here for the Thermal Rupture submodel. It has the further advantage of having been developed, debugged and validated at DAVIS.

BLEVE/Explosion Analysis

The BLEVE/Explosion submodel will have two aspects to it - a prediction of the onset of a BLEVE, and BLEVE or explosion consequences. Reid's work (2.6-3) is widely quoted as a standard BLEVE prediction technique, and will require programming for this application. The work of Leuschen and Birk (2.6-4) will be used to predict the explosion consequences. This will be done in the form of predicted loads on simple surrounding structures and by showing damage radii. These submodels could be supplemented by the texts by Kinney (2.6-9) and Baker (2.6-10).

Plume Dispersion

The code "SLAB" (2.7-4, 2.7-5, 2.7-6, 2.7-7, 2.7-8) is recommended for the heavy gas Plume Dispersion submodel. Although not as extensive as FEM3, (2.7-1, 2.7-2, 2.7-3, 2.7-4, 2.7-5), "SLAB" is only slightly less accurate, is far easier to execute computationally, and originates from the same research environment. It is available in the public domain, for a nominal processing fee. The other codes which were located were either outdated, or not available for the type of use required in this project.

For neutrally buoyant gas dispersion, it is recommended that the puff model of Peterson (2.7-11) be used. This model can be used to model the dispersion of instantaneous releases such as explosions and also continuous releases by using a continuous series of puffs.

3.0 SIMULATION SOFTWARE DESIGN

This chapter will deal with the design of the overall derailment simulation software. As mentioned previously, the program will be called "DERACS" for Derailment Accident Simulator.

The software will be developed in accordance with the recommendations made in Section 2.9. Table 3.0-1 presents a summary of the recommendations of Section 2.9 along with relevant information on the status of the software or models in question.

SUBMODEL	SOURCE	STATUS
Derailment Mechanics	- Yang and Manos in house	- to be further developed to allow for uncoupling to facilitate possibility of side on impacts
Impact Rupture	- Yin - Phillips - to be coded	- to be further developed to include side impact ruptures - impact model in the form of equations to be coded
Spill/ Release	- Fingas - "LEAKRATE" in house	- minor modifications for integration
Fire	- Crocker - "HOTPLT" in house - Birk	- Crocker analysis to be coded - minor modifications for integration - some fire effects to be handled by tank thermal model
Thermal Rupture	- Birk - "TANKCAR" in house	- Modifications to simplify for improved speed
BLEVE/ Explosion	- Reid - Leushcen and Birk in house	- Reid analysis to be coded - minor modifications for integration with other models
Plume Dispersion	- Morgan - "SLAB" to be procured - Peterson in house	- Minor modifications for integration - Puff model to be integrated other models

TABLE 3.0-1 RECOMMENDED MODELS TO BE USED IN SIMULATION SOFTWARE DEVELOPMENT

The above identified software that is not presently in house will be obtained. Based on the literature search, procurement of the above software will be within the budget constraints. It should be noted that all of the authors will be notified of the project intent to use their software.

There are no problems anticipated with obtaining the software. During a later task (software review) it will be determined whether the development requirements are in line with expectations.

With the various models identified, it was then possible to prepare a basic design for the overall simulation. The following sections present this design.

3.1 OVERVIEW OF SIMULATION APPROACH

The basic simulation approach will be as outlined in the original proposal to carry out this project. Figure 3.1-1 presents the overall accident simulator logic diagram as it was presented in the original proposal. The various submodels used for the simulation will be as described in previous sections. The following is a brief description of how the actual simulation will be performed. Discussion of the input and output requirements will follow.

The simulation will involve the calling of the main calculation subroutines in the following order (note that there are now additional submodels listed which will stem from those already discussed):

1) vehicle mechanics
2) impact rupture
3) BLEVE/explosion
4) explosion blast
5) explosion thermal
6) spill size
7) gas release rate
8) fire effects
9) plume track
10) puff track
11) fire impingement
12) blast rupture
13) thermal rupture
14) go to 3)

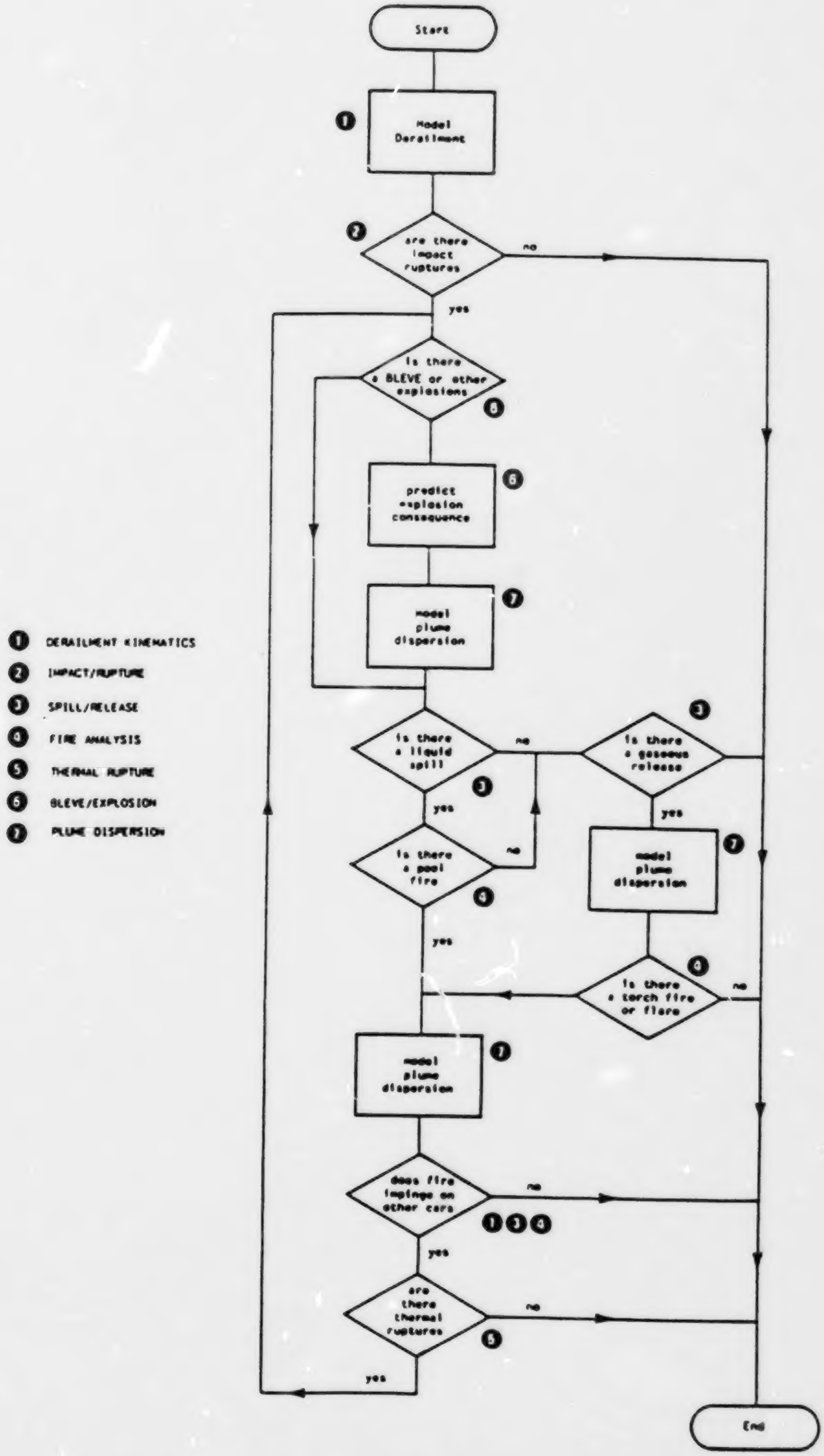

Start
① Model Derailment
② are there impact ruptures
no
yes
Is there a BLEVE or other explosions ⑥
predict explosion consequence ⑥
model plume dispersion ⑦
① DERAILMENT KINEMATICS
② IMPACT/RUPTURE
③ SPILL/RELEASE
④ FIRE ANALYSIS
⑤ THERMAL RUPTURE
⑥ BLEVE/EXPLOSION
⑦ PLUME DISPERSION
Is there a liquid spill ③
no
yes
Is there a gaseous release ③
yes
Is there a pool fire ④
no
yes
model plume dispersion ⑦
Is there a torch fire or flare ④
no
model plume dispersion ⑦
does fire impinge on other cars ①③④
no
yes
are there thermal ruptures ⑤
no
yes
End

FIGURE 3.1-1 OVERALL PROGRAM LOGIC

The simulation will involve calling the above subroutines for each car in the derailment. Markers will be set on each car to identify critical conditions such as whether it is exposed to fire, whether it has ruptured, whether it is leaking gas or liquid, etc. Each subroutine will only be called if a marker states that the car under consideration requires it. For those subroutines that are called to analyze spills, fires, etc., they will be called for each item separately. Items that will be treated individually the are:

1) cars
2) spills
3) releases
4) plumes
5) puffs
6) explosions
7) fires.

The first part will involve the simulation of the derailment mechanics. The actual calculation of the vehicle positions and the maximum forces will take a few minutes of computation time. The output of this submodel will be vehicle positions, rotations and the maximum forces applied to the vehicles. The mechanics routine will only be called once at the beginning of the simulation. The positions of the vehicles will then be displayed on the computer screen using the graphics format to be described later. The positions and forces will become inputs to the impact rupture submodel and the output of this will be the prediction of ruptures and their size and shape. The impact rupture submodel will require some development as this was identified as one area where active research is presently underway. It may be necessary to formulate a simple model for the prototype simulation.

If there are impact ruptures, the next submodel to be called will establish whether the commodity properties at the time of rupture were sufficient to cause a BLEVE or a fireball type explosion. If there is an explosion or BLEVE, the car will be marked appropriately so that when the explosion affect routines are called they will come into action.

For the cars that explode, the explosion blast and thermal subroutines will calculate the hazard levels and their centres and sizes will be stored for display. A puff marker will also be allocated so that the puff track can be plotted.

If there is commodity left to leak from the tank, then spill and gaseous leak rates will be estimated based on the tank pressure, the commodity temperatures and the estimated size of the tank ruptures. Once again, this information will be stored for each tank and will remain stored until needed by a subsequent subroutine.

If the commodities that are released either as gases or liquids are flammable, then it will be assumed that they are somehow ignited. Therefore, flammable pools will be pool fires and flammable flares will be burning flares. The fire effects routine will also calculate burning rates so that burn out times can be predicted. When all of the fuel is consumed, the fire will go out. Appropriate markers will be set to indicate these fires, their grid locations and their sizes.

Volatile spills and gaseous releases will be tracked using the plume dispersion model. The origin and size of the plume for a predetermined threshold concentration will be stored for display. Each spill and release will be treated individually. Explosions clouds will be tracked by a puff model in a similar fashion.

Fire impingement on vehicles will be determined by comparing the grid locations of vehicles with the locations of known fires and flares. Each fire contact will be treated as a unique fire impingement. Torch fires will be modelled as such with an effective torch temperature and area based on the distance to the target from the torch source. Engulfing or partially engulfing fires will be treated as totally engulfing fires but with different effective fire temperatures. Fire impingement temperatures and areas will be stored for later use by the tank-car thermal model for prediction of thermal rupture.

The final submodels to be called are used to predict blast and thermally induced ruptures. The blast rupture prediction will be based on the output from the explosion blast effects submodel. Any cars within a severe blast damage radius will be marked as being ruptured due to explosion blast effects. The size of the blast ruptures will be assigned based on the proximity of the blast centre. The thermal rupture prediction will be carried out using the tank-car thermal model TANKCAR. This subroutine, as with the others, will be called for every car that is in contact with fire as indicated by the individual car makers. Individually stored data on the cars will also indicate the relevant data such as fire temperature, fire contact area, current tank fill, etc. If "TANKCAR" indicates a thermal rupture this too will be stored as a marker for that car for consideration the next time the BLEVE/explosion routine is called. Model "TANKCAR" will also indicate if a tank has emptied safely.

At the end of the TANKCAR predictions, the entire calculation sequence will repeat. The repetitions will continue until all the fires are out, or all the tank-cars are empty or destroyed.

Preparing for, and conducting a simulation will involve four distinct activities:

1) user setup of simulation boundary and initial conditions which will involve one of the following three possibilities:
   a) the input of the characteristics of the train consist (number, order and type of vehicles, commodities, speed) characteristics of the rail track (curvature), characteristics of surroundings including position of buildings and roads, wind conditions, etc., and the initiation point of the derailment (position on the track and the identification of the derailment leader (i.e. the car that derails first)), or
   b) as in a) above except that the derailment simulation is bypassed and the accident scene, including vehicle placement, location of ruptures, spills, fires, releases, etc. are defined as input, or
   c) default input (sample or demonstration simulation) or existing input file from previous pre-processor session.
2) Performance of the simulation which will involve the time solution of the derailment and its probable consequences or for 1 b) above the time solution of the probable consequences of a defined derailment scene.
3) The graphical presentation of the derailment simulation as it is being generated by the computer including plan views of the accident

scene and

4) a text summary of the simulation events such as ruptures, explosions, and likely property damage along with the times of these events from the initiation of the derailment simulation.

Each of these phases will be described in more detail in the sections that follow. The first phase which is the input phase will be done using a preprocessor. The second part which is the actual simulation calculation and will be conducted by the various submodels discussed in previous sections. A brief description of how these submodels will work together also follows. The third and fourth parts comprise the simulation post processor and will be described in further detail in a following section.

3.2 INPUT PREPROCESSOR

The input preprocessor will be menu driven. The total number of inputs for the present simulation will be kept to a minimum for the prototype software but the software will be engineered so that future development activities to enhance the capabilities are not restricted. Note that this is a preliminary design and further refinements, changes, and deletions are anticipated. Also note that there will be a limited number of input options available for this prototype simulation package. Additional options can be added in the future.

The preprocessor will begin by asking the user what type of input is desired. The question will follow the questioning logic presented in Figure 3.2-1.

Question	Options	Comments
Input type	1) Default to demo	- Predefined demonstration simulation
	2) Existing Input File	- Previously set up by the User
	3) Full Input for Simulation	
	4) Full Input for Simulation without Derailment Mechanics	

FIGURE 3.2-1 PREPROCESSOR QUESTIONING LOGIC: INPUT TYPE

If the selected option is 1) or 2) then the simulation will proceed by loading in the necessary input files and the simulation will begin. The input data files will be organized in a logical fashion so that an experienced user can enter the file to make changes (for example, changing the train speed to see what effect this has on the probably consequences of the derailment). Note that changing of an existing data file will not be an option in the preprocessor and therefore will not be a menu directed activity.

If option 3) is selected, then the preprocessor will prompt the user to enter the relevant data as shown in Figure 3.2-2. If option 4) is selected, then the preprocessor will prompt the user for data as shown in Figure 3.2-3. Upon completion of the preprocessing, the input data is stored and the simulation will begin. In both cases, the input is organized into three or four input data packets (whichever is required depending on the type of simulation requested) as follows:

a) train consist data

- number of cars
- type of cars
- thermal protection (insulation)
- commodities
- train speed
- car, truck, axle that initiates derailment

b) ground surroundings data

- track (single, double)
- roads (large, medium, small)
- buildings (cylinders, square, rectangles)
- property boundaries
- water ways or bodies
- hills (cones, wedges)
- forest
- point on track where derailment initiates

c) atmospheric conditions

- wind speed and direction
- temperature
- atmospheric conditions (stability class, etc.)

d) derailment scene data:

- position and roll angle of vehicles
- location and descriptions of ruptures
- location and descriptions of spills
- location and descriptions of fires
- location and descriptions of explosions.

The input data is formulated based on a rectangular coordinate system for the 'world'. The world will consist of a 10 km x 10 km square subdivided into 10,000 square each measuring 100 m on a side. Thus, the world consists of a 100 x 100 matrix of square 100 m x 100 m. An item is placed by stating its coordinates and rotation (for example, a vehicle is placed by locating the x and y coordinates of the vehicle centre and the rotation angle of the car (measured counterclockwise from the x axis).

QUESTION	OPTIONS	COMMENTS
Train Consist Data		
Number of cars	1-40 cars	- the final number will depend on hardware limitations
Type of Cars	1) Locomotives 2) Tank-Cars 3) Others	- other car types are to be defined
Thermal Protection	1) Insulation	- applies to tank cars only
Commodities	1) Propane 2) Chlorine	- applies to tank cars only
Train Speed	0 - 100 km/hr	
Derailment Leader		
Car	1 - 40	
Truck	1 or 2	
Axle	1 or 2	

FIGURE 3.2-2 PREPROCESSOR QUESTIONING LOGIC: (INPUT TYPE 3) (INPUT FOR FULL SIMULATION)

QUESTION	OPTIONS	COMMENTS
ground surroundings data		
Track	yes - no	- if no, go to roads
Type	single double	
Type	tangent curved	
Location	x, y to x, y	
Roads	yes - no	- if no, go to buildings
Type	large medium small	
Type	tangent curve	
Location	x,y to x,y	
Buildings	yes - no	- if no, go to boundaries
Type	cylinders squares rectangles	
Size	l,w,h or r,h	
Location	x,y,θ or x,y	

FIGURE 3.2-2 PREPROCESSOR QUESTIONING LOGIC: (INPUT TYPE 3) (INPUT FOR FULL SIMULATION) (continued)

QUESTION	OPTIONS	COMMENTS
Boundaries	yes - no	- if no, go to water
Line	x,y to x,y	
Water Ways or Bodies	yes - no	- if no, go to hills
Patch Type	square circle rectangle	
Patch Size	l,w or r	
Location	x,y,θ	
Hills	yes - no	- if no, go to forests
Type	cones wedges	
Size	b,h,l	
Location	x,y,θ	
Forest	yes - no	- if no, go to more input
Patch Type	square circle rectangle	
Patch Size	l,w or r	
Location	x,y,θ	

FIGURE 3.2-2 PREPROCESSOR QUESTIONING LOGIC: (INPUT TYPE 3) (INPUT FOR FULL SIMULATION) (continued)

QUESTION	OPTIONS	COMMENTS
More data	yes - no	- if no, go to point of derailment
Type	cars track road buildings boundaries water hills forest	
Point on track where derailment initiates	x,y	
atmospheric conditions		
wind speed	0 - 100 km/hr	- limited by
wind direction	0 - 360 degrees	- measured ccw from x axis
temperature	-40 - 40 deg C	-
atmospheric conditions	________________	- requirements to be determined from code used

FIGURE 3.2-2 PREPROCESSOR QUESTIONING LOGIC: (INPUT TYPE 3) (INPUT FOR FULL SIMULATION) (continued)

QUESTION	OPTIONS	COMMENTS
train consist data		
number of cars	1-40 cars	
type of cars	1) locomotives 2) tank-cars 3) others	- other car types are to be defined
thermal protection	1) insulation	- applies to tank cars only
commodities	1) propane 2) chlorine	- applies to tank cars only
ground surroundings data		
track	yes - no	- if no, go to roads
type	single double	
type	tangent curved	
location	x,y to x,y	

FIGURE 3.2-3 PREPROCESSOR QUESTIONING LOGIC: (INPUT TYPE 4) (INPUT FOR SIMULATION WHERE DERAILMENT MECHANICS SIMULATION IS BYPASSED)

QUESTION	OPTIONS	COMMENTS
train consist data		
roads	yes - no	- if no, go to buildings
type	large medium small	
type	tangent curve	
location	x,y to x,y	
buildings	yes - no	- if no, go to boundaries
type	cylinders squares rectangles	
size	l,w,h or r,h	
location	x,y,θ or x,y	

FIGURE 3.2-3 PREPROCESSOR QUESTIONING LOGIC: (INPUT TYPE 4) (INPUT FOR SIMULATION WHERE DERAILMENT MECHANICS SIMULATION IS BYPASSED) (continued)

QUESTION	OPTIONS	COMMENTS
boundaries	yes - no	- if no, go to water
line	x,y to x,y	
water ways or bodies	yes - no	- if no, go to hills
patch type	square circle rectangle	
patch size	l,w or r	
location	x,y,0	
hills	yes - no	- if no, go to forests
type	cones wedges	
size	b,h,l	
location	x,y,θ	
forest	yes - no	- if no, go to more input
patch type	square circle rectangle	
patch size	l,w or r	
location	x,y,θ	

FIGURE 3.2-3 PREPROCESSOR QUESTIONING LOGIC: (INPUT TYPE 4) (INPUT FOR SIMULATION WHERE DERAILMENT MECHANICS SIMULATION IS BYPASSED) (continued)

QUESTION	OPTIONS	COMMENTS
more data	yes - no	- if no, go to point of derailment
type	cars track road buildings boundaries water hills forest	
atmospheric conditions		
wind speed	0 - 100 km/hr	- limited by
wind direction	0 - 360 degrees	- measured ccw from x axis
temperature	-40 - 40 deg. C	-
atmospheric conditions	______________	- requirements to be determined from code used

FIGURE 3.2-3 **PREPROCESSOR QUESTIONING LOGIC: (INPUT TYPE 4) (INPUT FOR SIMULATION WHERE DERAILMENT MECHANICS SIMULATION IS BYPASSED) (continued)**

QUESTION	OPTIONS	COMMENTS
derailment scene data:		
car number	1 - 40	
coupled	yes - no	
on track	yes - no	
location	x,y,θ	
roll	0 - 360 deg.	
ruptured	yes - no	- applies to tank-cars only
type	round hole Jagged hole long tear lost end jacknife	- allowable geometries as required by LEAKRATE
size	d,w,l etc.	
burning flare	yes - no	
spills	yes - no	- if no, go to fires
size	d	
location	x,y	
flammable	yes - no	
burning	yes - no	

FIGURE 3.2-3 PREPROCESSOR QUESTIONING LOGIC: (INPUT TYPE 4) (INPUT FOR SIMULATION WHERE DERAILMENT MECHANICS SIMULATION IS BYPASSED) (continued)

QUESTION	OPTIONS	COMMENTS
explosions	yes - no	- if no, go to more
location	x,y	
damage index	1 - 10	- damage index scale to be determined
more	yes - no	
car number spills fires explosions		

FIGURE 3.2-3 **PREPROCESSOR QUESTIONING LOGIC: (INPUT TYPE 4) (INPUT FOR SIMULATION WHERE DERAILMENT MECHANICS SIMULATION IS BYPASSED) (continued)**

Note that with the input logic as shown the user cannot correct errors while using the preprocessor. To correct an error the user must enter the final data file and must delete the erroneous line of information. An error correction feature may be added later in the project if budget constraints allow it.

During the development of the preprocessor other methods of data input may be developed which improve its efficiency. For example, the proposed methods for locating bodies of water and forests use a patching technique where the patches will be simple shapes. It is possible to do this in other ways and these will also be considered during the preprocessor development.

As stated earlier, the preprocessor session will end with the input data being stored in a file on disk.

3.3 SIMULATION SOFTWARE DEVELOPMENT

The simulation software will be based on the submodels described in Section 2.9. Figure 3.3-1 presents a flow chart showing the overall program flow as it was presented in the proposal for this study. The basic simulation design remains the same but now the diagram shows the actual software or models that will be used. As stated previously, these submodels will be:

- derailment mechanics
- impact rupture
- spill and releases
- fires
- thermal ruptures
- BLEVES/Explosions
- plume and puff dispersion.

The selected codes and models have been summarized in Section 3.0. Each of the above will require development so that they can work together as the derailment simulator. Some will require simplification. Those models not presently coded will be put into code form. In those few cases where models do not exist, simple models will be formulated and coded. Then all codes will be integrated and will be merged with the pre and post processors.

Development requirements for the above submodels will now be briefly presented individually. As with all the preliminary design items addressed in this report, details may change as the model development progresses.

Derailment Mechanics

The simulation of derailment mechanics will follow closely the analysis of Yang et al [2.1-1, -2]. Most of the important mechanical aspects of the derailments have been considered by them and are of use in the development of this simulation. However, as outlined in previous sections, it is considered important that the model be further developed to include the effects of tank roll and car uncoupling. This, therefore, is one submodel requiring a significant level of development.

A derailing vehicle is acted upon by relatively few, though complex, forces. Immediately after derailment, the car will be assumed to lose contact with the rails so that it will begin reacting to forces of friction between the soil and the wheels. Yang et al. have modelled these forces as being proportional to the total normal force acting between each truck and the ground and acting in a direction which directly opposes the truck motion. The present model will include a similar description of the forces, but will include the overturning moments caused by these forces.

In this model, gravitational forces become more important than they were in the model of Yang et al. There, they were used only to calculate the normal forces on the trucks so that the resulting soil friction force could be calculated. Here, they still appear in the soil interaction forces but will also be used to determine, in conjunction with the lateral component of the soil friction force, the net overturning moment acting on the vehicle. As the vehicle leaves the rails and moves into a three-dimensional position on the way-side, there will be a component of the gravitational force tending to overturn the railcar.

Coupler forces will be modelled in more detail than they were by Yang et al. The model of Ward and Leonard [2.1-4], supplemented by a limiting breakage force, will be used. Once a coupler is predicted to have been broken, the neighbouring cars will be freed from each other, resulting in the necessity to track the progress of individual cars or short chains of cars. Such a calculation is much more complex than has been treated in the past, but adds realism to the simulation and gives the potential for predicting tank car punctures resulting from impact with free couplers.

Impact Rupture

At present, there has not been a specific code or model identified for this submodel. Therefore, a model for impact rupture will be developed based on the references identified in the literature search. This, therefore, is one submodel identified as requiring some development. The level of effort required to develop a simple model is considered to be within the scope of the present project.

The impact rupture analysis will be based on the following logic. A three-dimensional envelope around each car will be established and checked at each step of the derailment simulation for intrusion from other cars. Should a collision be predicted, the resulting forces at the impact point will be calculated from the application of a constraint equation which does not permit one body to enter another. The resulting impact forces will be tested against an established puncture criterion (Section 2.2) to determine the location of damaged cars. For each damaged car, the impact forces and angles will be used to estimate the size, location and shape of the resulting puncture. For example, the "LEAKRATE" program allows for round or rectangular puncture that are either clean or with jagged metal bent inward or outward. For impacts due to coupler intrusion, it would be reasonable to assume rectangular holes with jagged metal bent inward. Release rate would then be predicted based on this assumption.

Spills and Releases

The LEAKRATE program requires the following items as input:

- commodity
- commodity and ambient temperature
- ambient pressure
- wind speed
- tank geometry and orientation
- tank size and wall thickness
- properties of thermal insulation if present
- rupture type, size and location
- mass of commodity in tank.

All of the above information will be available from either the overall "DERACS" input or from other submodel outputs. From these inputs, "LEAKRATE" calculates release rates of liquids or vapours whichever is appropriate. These release rates will then be used to predict spill areas, plume dispersion, etc. Estimates of evaporation rates from spills will be based on simple heat transfer considerations and on published burning rates for pools if there is a fire present. During every timestep, the inputs to "LEAKRATE" will be updated and release rate estimates will be recalculated. Eventually, the spill or release source will be depleted.

Except for program integration and simplification (LEAKRATE also predicts the time for the source to be depleted but since this will be done by other submodels it is no longer needed in "LEAKRATE" no major program development is anticipated for the spill and release rate submodel. At present, "LEAKRATE" does not include propane as a commodity and therefore it will be added.

Fires

The effects of fires will be considered in a number of ways in the "DERACS" simulation, including:

- pool fires impinging on tank-cars
- burning flares impinging on tank-cars
- fire balls from explosions and their effects on neighbouring structures
- large pool fires and their effects on neighbouring tanks and structures.

Fire impingement on tanks will be handled by the tank thermal rupture analysis since this is presently part of that submodel. For the thermal rupture analysis to be carried out it will be necessary to define the properties of the torch or pool fire and this is done by specifying the degree of exposure of the tank (fully engulfed, partially engulfed, or size of torch and distance to source, etc.). Since the thermal model presently handles only fully engulfing or torch type fires the case of a partially engulfing fire will be handled by using the effective fire temperatures (i.e. a partially engulfing fire will be modelled by using a typical engulfing fire temperature for predicting wall temperatures but a reduced flame temperature for the thermodynamics process calculation (i.e.g tank pressures lading temperatures)).

For fire effects from large pools or fireballs, the analysis methods of Crocker (2.4-8) will be used as described earlier. It should be noted that most of these models are based on fundamental radiant heat transfer considerations and therefore are well documented. These methods are well defined in the referenced report and should be straightforward to code. Therefore, except for the coding and integration of these models, no significant model development will be required here.

Thermal Ruptures

As stated earlier, the thermal rupture submodel will be based on the Tank-Car Thermal Model "TANKCAR" developed by Birk. For the purposes of this application, "TANKCAR" will be modified to speed the program and to adapt it to be able to track the progress of a maximum of eight tank cars exposed to fires. The modifications will include the following:

1) removal of unwanted parts of the program including:
    - advanced thermal protection models for such things as EXPLOSAFE, fusible plugs, internal shells.
2) simplification of tank representation including:
    - reduced number of nodes in the tank heat transfer and temperature distribution submodels.
3) reducing run time by:
    - replacing certain calculations (thermodynamic properties) by table look-ups
    - relaxation of convergency tolerances in iterative calculations.
4) expansion of vectors and matrices to allow for the simultaneous tracking of eight tank-cars exposed to various levels of fire impingement.

Once the modifications have been conducted, the modified TANKCAR will be validated against fire test data for a full scale unprotected tank and a full scale thermally insulated tank. This part of the overall simulation program will require no input other than that already discussed in section 3.2.

BLEVES/Explosions

As stated earlier, this submodel will be made up from the methods of Reid (2.6-3) for the prediction of the onset of a BLEVE and the program of Leuschen and Birk for prediction of blast effects. Since the models are already developed only the program coding of Reid's work remains to be carried out.

For the prediction of explosion blast effects, the Program BLAST by Leuschen and Birk (2.6-4) will be used. For explosion thermal effects the same submodels developed for fire effects will be used (based on the analysis of Crocker, etc.).

The program "BLAST" presently requires the input of the following:

- type of explosive
- efficiency of the explosion
- mass of explosive
- distance to the nearest face of an exposed structure
- size and shape of the structure.

The above required inputs are, except for the explosion efficiency, all available from various submodels and inputs to the "DERACS" program. A simple model to establish explosion efficiencies based on the condition appropriate to the simulation will be developed. The explosion efficiencies will be based on data in the open literature for the types of explosions to be expected in this application.

From these inputs, "BLAST" calculates the free field overpressure and the resulting forces applied to the sides of the structure, and a damage estimate for a range of distances. These are the outputs required for the "DERACS" program and therefore very little effort will be required to integrate "BLAST" into the "DERACS" program.

Plume and Puff Dispersion

Propane and chlorine are examples of heavy gases. In the present work, it is important to model the dispersion of these gases if they are released. As stated in Section 2.9, it is recommended that the computer program "SLAB" by Morgan (2.7-4) be used for predicting the dispersion of these commodities.

In the case of burning flares and pool fires, it is also of interest to plot the trajectory of these releases even though they may not be of a toxic nature. These plumes will initially be buoyant because of their elevated temperatures but eventually they will cool and become neutrally or negatively buoyant. For the present application, it will be assumed that they behave as neutrally buoyant so that they can be tracked using relatively simple models. It has been recommended that the Gaussian puff model as coded by Peterson (2.7-11) be used for this purpose.

The plume and puff models will track the movement of gaseous releases. Explosions will track the movement of gaseous while flares, leaks and fire plumes will be treated as continuous (or as a series of puffs depending on whether the gas released is neutrally buoyant or heavier than air). The plume or puff tracks will be plotted as a function of time on the computer screen as the DERACS simulation is carried out.

The program "SLAB" is based on the solution of the governing equations for mass, momentum, energy and species in one space dimension and time. The authors call the analysis quasi-3-D because the program predicts the concentration, and dimensions of the cloud with downwind distance vs time. These outputs are all that is required for the present application. "SLAB" requires the following as input:

- source size
- initial cloud dimensions
- source gas properties (T, density, etc.)
- atmospheric conditions including wind speed and profile, entrainment characteristics, temperature, etc.
- ground characteristics including friction coefficients, temperature, etc.

The source properties will of course be outputs of other submodels in "DERACS". The various atmospheric and ground properties will have to be inputs. Many of these inputs can be fixed for the purposes of the prototype simulation and will be based on the guidelines presented in the various references on "SLAB".

The SLAB program will be more difficult to integrate with the rest of the DERACS submodels because of its complexity. However, no problems are anticipated. It is simply a matter of expending additional review time to understand the internal workings of SLAB so that it can be modified to fit the needs of the present application.

The puff model of Peterson as mentioned in Section 2.7 is based on the Gaussian puff equation. This equation gives the peak surface concentration from an instantaneous surface release of a neutrally buoyant gas. This analysis is appropriate for release times where the meteorological conditions (wind speed, atmospheric stability, etc.) remain steady. It is considered to be a valuable tool for estimating toxic corridors in accidents and is therefore considered appropriate for the present application.

This model can give the peak gas concentration as function of downwind distance. The program requires the following as input:

- source strength
- source height
- atmospheric dispersion parameters
- atmospheric stability class
- wind speed

All of the above will be either inputs to DERACS or calculated by other submodels. The atmospheric parameters will be asked for in the preprocessor and the guidelines for these parameters will be used on those presented by Peterson (2.7-11).

The Gaussian puff model is not a transient model. That is, it presents the peak concentrations with distance from the source, not the time varying concentrations. In "DERACS", the centre of the moving puff cloud will be located by considering the wind speed and direction. The peak ground concentrations resulting from the puff moving over will be calculated and presented in the simulation. After a time step, the new puff location will be calculated and newly effected ground will be indicated but previously effected ground will not have their concentrations recalculated.

No development is required for the Gaussian puff model other than its integration with the rest of the "DERACS" simulation software.

During the testing of the "DERACS" program, it will be determined whether it is necessary to have both the SLAB and the puff model. it may be decided at that stage that only one of the models is necessary for the purposes of the prototype simulation software. More advanced plume and puff dispersion models can be added at a later date if it is deemed necessary.

3.4 POSTPROCESSOR

As stated earlier, the postprocessor will be used to generate the graphic display as the simulation is running and will also prepare a graphic and text summary of the simulation events for printing in hard copy. During the simulation, the user will be able to ask for a hard copy snapshot of the screen for hardcopy. The text and graphic summary will include a plan view image of the initial derailment scene and the text summary will include the following:

- date, time, location or title of derailment
- time, grid locations affected, event (rupture, explosion, fire, empty, etc.), commodity involved, and cars affected.

The above list will be modified at the postprocessor is developed. For events that evolve with time, such as gaseous releases, the text summary will give the start time of the event and will summarize the overall effect. Figure 3.4-1 presents a sample of how the text summary may look for a simulation. The text summary will be generated as the events occur and therefore the events will be printed out in the order in which they happened.

The graphic display will be designed to operate on a monochrome display but this may prove difficult for the number of items that will be shown. Therefore, it may be necessary to generate the graphics in colour. The hardcopy summaries will be in black and white. The graphic display will show the following events as they progress through the simulation:

- vehicle placement
- surroundings
- compass, wind direction and speed in vector form
- ruptured vehicles (impact induced)
- ruptured vehicles (thermally induced)
- spills
- pool and torch fires
- vehicles exposed to fires
- plume tracks
- puff tracks
- explosion thermal effect zones
- explosion shock effect zones.

The various effect zones will be indicated by cocentric circles centred at the explosion site. The different circles will show distances to different levels of

DERACS: Derailment Accident Simulator - Text Run Summary

time grids affected title:

TIME	GRIDS AFFECTED	EVENT	COMMODITY	CARS AFFECTED
00:01:12	A2	impact ruptures	propane	6,7
00:13:33	A2, 3	spill	propane	6,7
00:13:33	A2,3	fire	propane	11,14,22,24
00:16:11	A2	thermal rupture	propane	13
00:16:12	A2,3,4,5,6 B3,4,5,8,9 C1,4,7,8,9,10	BLEVE	propane	14
00:16:15	A2,3	blast rupture	chlorine	8
00:16:22	A2,3,5 B2,3,4,5,6,7 C4,5,6,7,8,9,10 D5,6,7,8,9	release	chlorine	8

FIGURE 3.4-1 SAMPLE OF PROPOSED TEXT SUMMARY FOR SIMULATION

effect. For example, for the thermal effect zones the levels of effects will be defined in terms of the thermal radiant power levels required to ignite different materials and also the effects on human beings. The shock effect zones will be similar except the zones will be defined in terms of blast wave damage on buildings, machinery and humans.

When viewing the accident on the computer screen, three levels of view will be available, world (showing 10 km x 10 km scene), local (showing 1 km x 1 km scene) and detail (showing 100 m x 100 m scene). The method of change views will be described later.

Figure 3.4-2 presents a plan view of an accident scene as it might look in detail view just following the derailment mechanics simulation. The graphics key is shown in Figure 3.4-3. The user will be able to view this graphics key by pressing the appropriate function key as will be described later. The planned images may vary from those shown depending on hardware limitations. As was mentioned earlier, if the images become too difficult to interpret it may be necessary to use colour which of course will affect the way different items are represented.

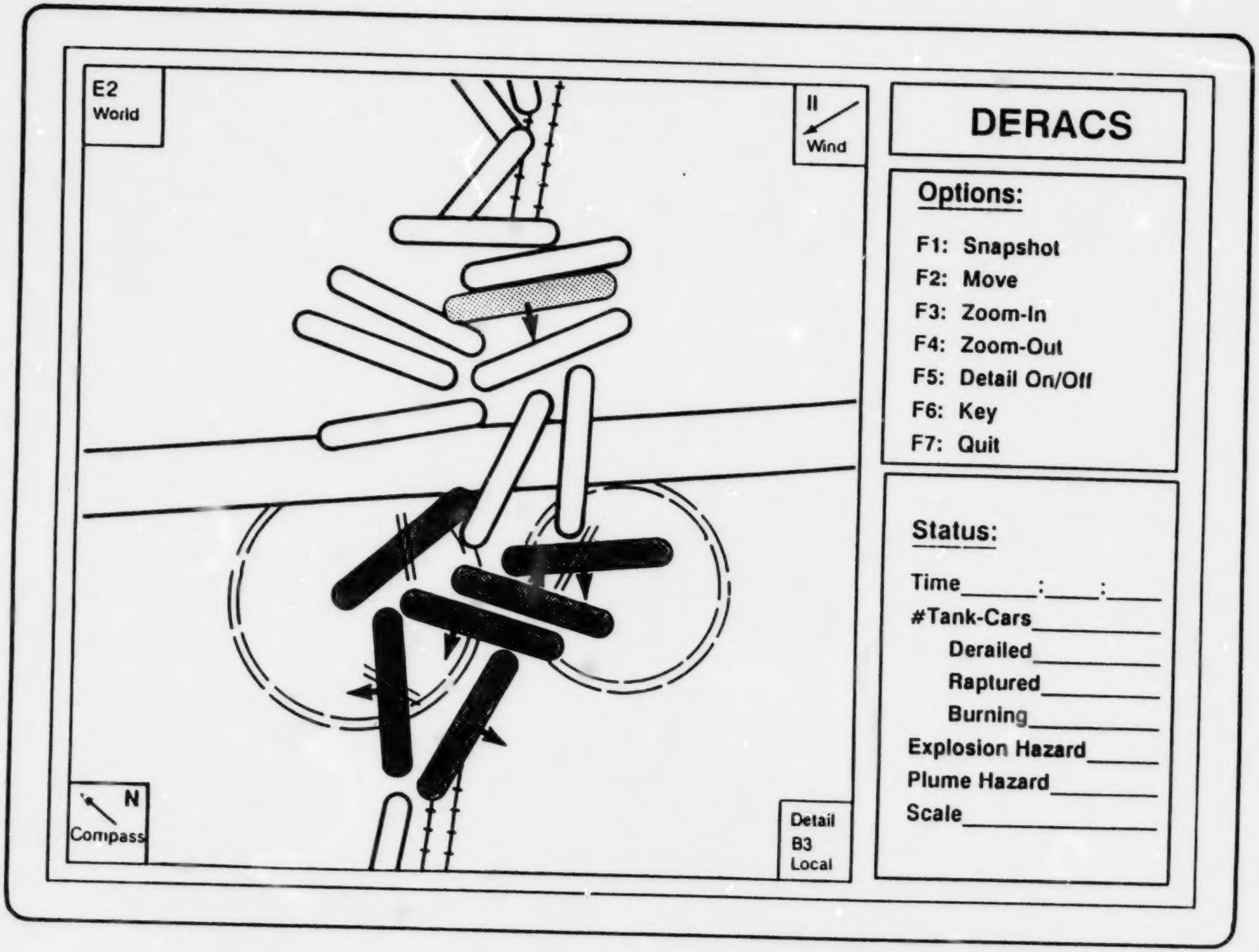

E2
World
II
Wind
DERACS
Options:
F1: Snapshot
F2: Move
F3: Zoom-In
F4: Zoom-Out
F5: Detail On/Off
F6: Key
F7: Quit
Status:
Time_____:____:____
#Tank-Cars_____
Derailed_____
Raptured_____
Burning_____
Explosion Hazard_____
Plume Hazard_____
Scale_____
N
Compass
Detail
B3
Local

FIGURE 3.4-2 ACCIDENT VIEW AT DETAIL LEVEL OF SCALE (NOT TO SCALE)

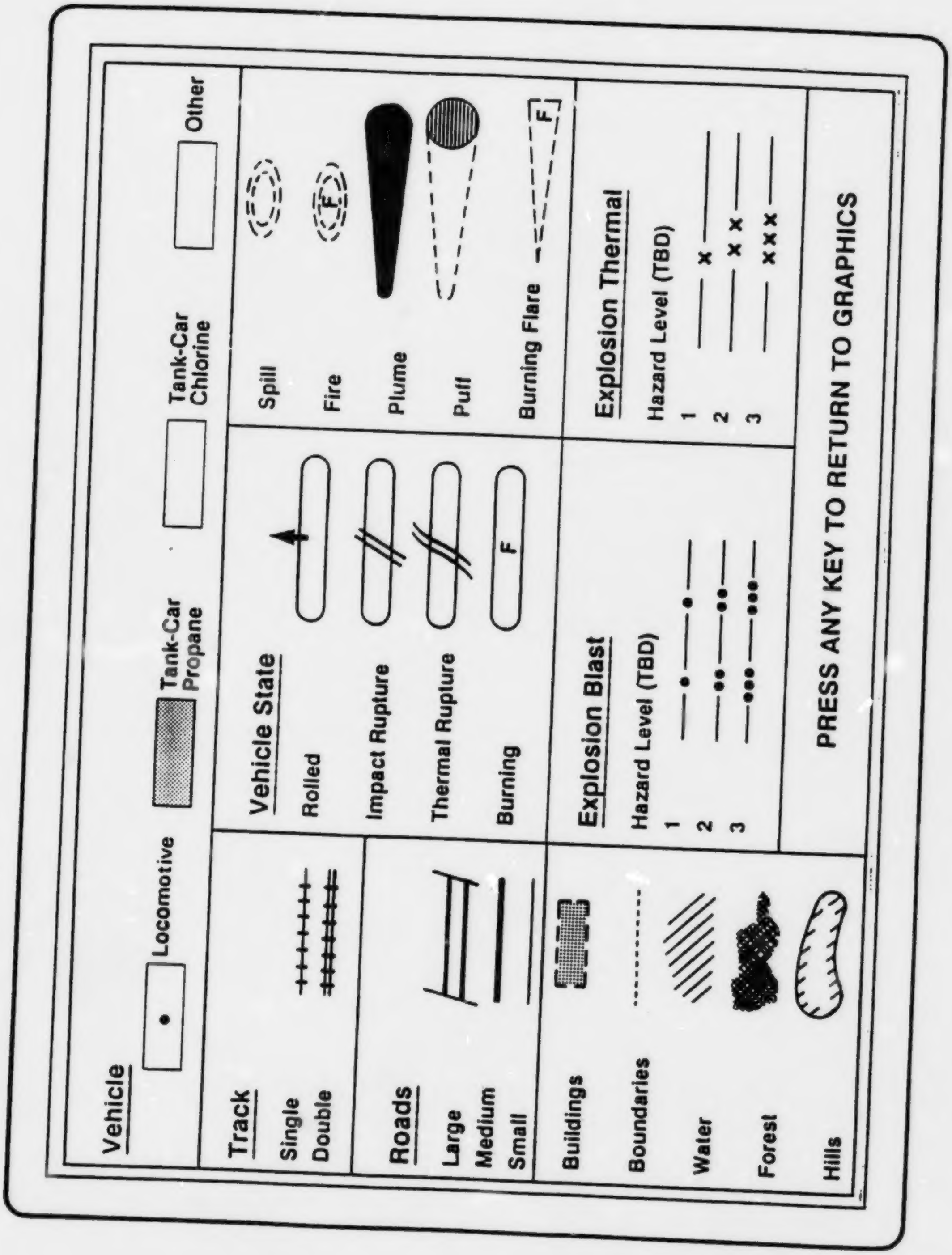

Vehicle
Locomotive
Tank-Car Propane
Tank-Car Chlorine
Other
Track
Single
Double
Roads
Large
Medium
Small
Buildings
Boundaries
Water
Forest
Hills
Vehicle State
Rolled
Impact Rupture
Thermal Rupture
Burning
F
Explosion Blast
Hazard Level (TBD)
1
2
3
Spill
Fire
F
Plume
Puff
Burning Flare
F
Explosion Thermal
Hazard Level (TBD)
1
2
3
x
x x
x x x
PRESS ANY KEY TO RETURN TO GRAPHICS

FIGURE 3.4-3 GRAPHICS KEY PAGE

Note the cars that have been identified as tank-cars and which of those have been ruptured by impacts. Also, note the spills that have accumulated around the ruptured cars and which of these are on fire, and the tank-cars that are exposed to these fires.

Figure 3.4-2 also shows how the computer screen will look including the run time options available to the user. These options will be invoked by pressing the appropriate function key. The options will be as described below:

F1: SNAPSHOT - by pressing this option the user stores the current graphical display for later printout on a dot matrix graphics printer. The data stored is exactly what is shown on the screen at the time the function key is pressed.

F2: MOVE - when this key is pressed, the user wants to move to another grid location. After this key is pressed, the display will change to show the world including grid numbers as shown in Figure 3.3-4. At this time the user will be asked to enter the coordinates of the grid that is to move to the centre of the computer screen. Once this is done, the user will be returned to the original scale with the new grid location centred on the screen.

F3: ZOOM IN - when this key is pressed, the scale will change on level of magnification to increase the scene resolution. The current grid will remain in the centre of the screen. The finest resolution will be when one of the 100m x 100m grids fills the screen. The coarsest resolution will be when the full world (10 km x 10 km) fills the screen.

F4: ZOOM OUT - when this key is pressed, the scale will change one level of magnification to decrease the scene resolution. The current grid will remain in the centre of the screen.

F5: DETAIL ON/OFF - when the user presses this function key the scene detail level is toggled. When the detail toggle is ON the graphic scene will show everything. When the detail toggle is off, the screen will only show the surroundings, vehicle placement, vehicle status, spills, releases, and fires. Explosion effects and plume and puff tracks will not be shown.

F6: KEY - when the user presses this key, the screen will show the graphics key page (see Figure 3.4-3). To return to the simulation graphics, the user must press any key on the keyboard.

F7: QUIT - when the user presses this option, the simulation will end. The events up until that time will be summarized in graph and text form.

Figure 3.5-2 also shows the way the graphics display will summarize critical simulation data such as time and number of cars involved in the fires, etc. The final format of the graphics page will evolve and its design will be dictated by available space and hardware capabilities. It is possible that an additional text or graphics page may be used to summarize simulation data and that this page could be called up using another function key.

Figure 3.4-5 presents a plan view of the accident scene the way it might look later on in the accident. This figure shows a scene at the local level (1 km x 1 km). Note that one of the tank-cars exposed to fire has ruptured and the resultant fireball has affected a neighbouring building as indicated by both the shock and thermal loading lines. The explosion cloud or puff is indicated in the view. The plumes resulting from the fires and flares is shown.

Figure 3.4-6 presents a plan view of the accident at the world level (10 km x 10 km). From this view, the overall picture takes form. The explosions have only affected a small area but the plumes are dispersing over a wide area.

As stated above, the user has the ability to record any of the above views as the simulation is taking place. During the development of the graphics preprocessor, every attempt will be made to keep the views as clear and as understandable as possible. However, the graphics will be hardware limited in this application and therefore some simplifications from the above plan may be necessary.

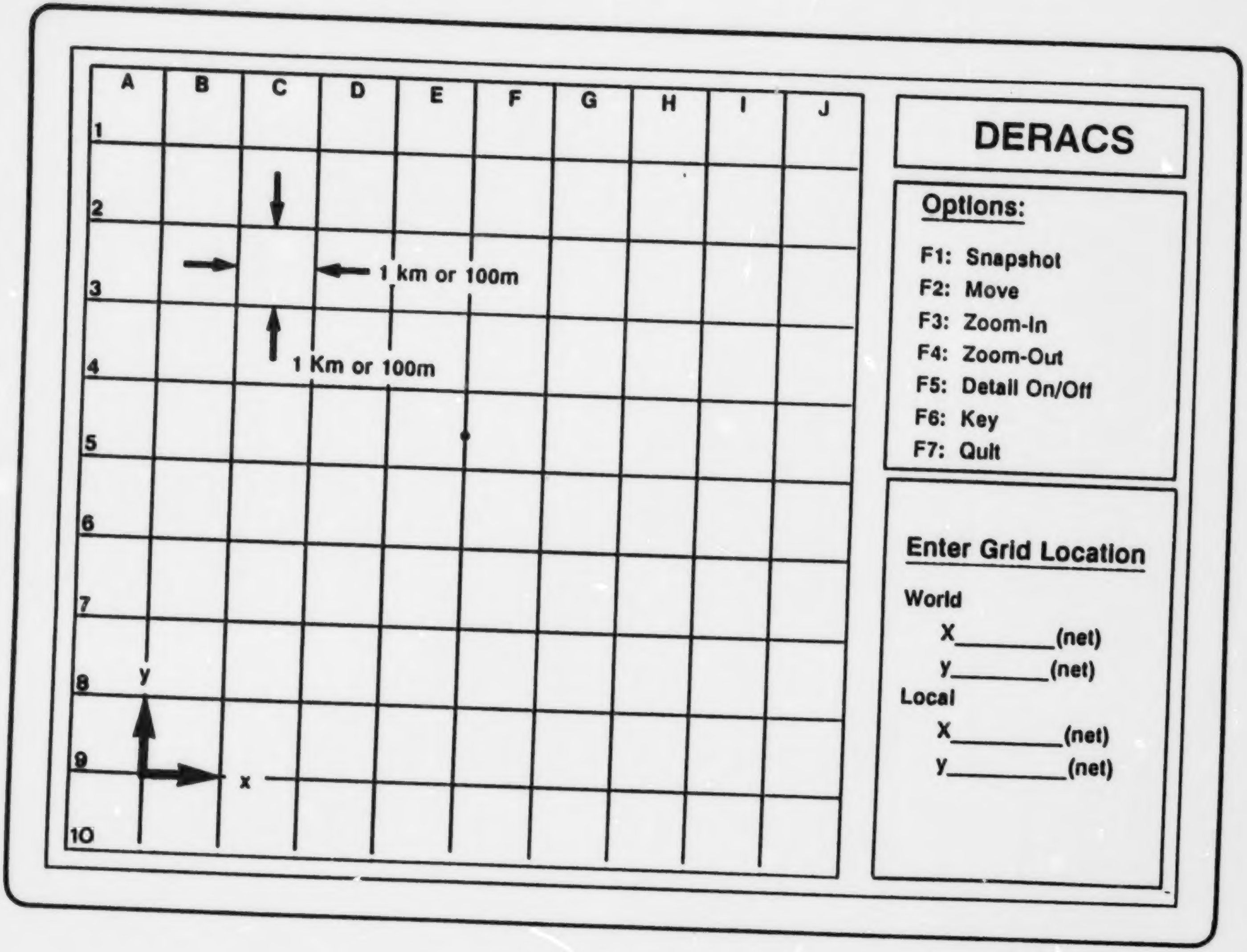
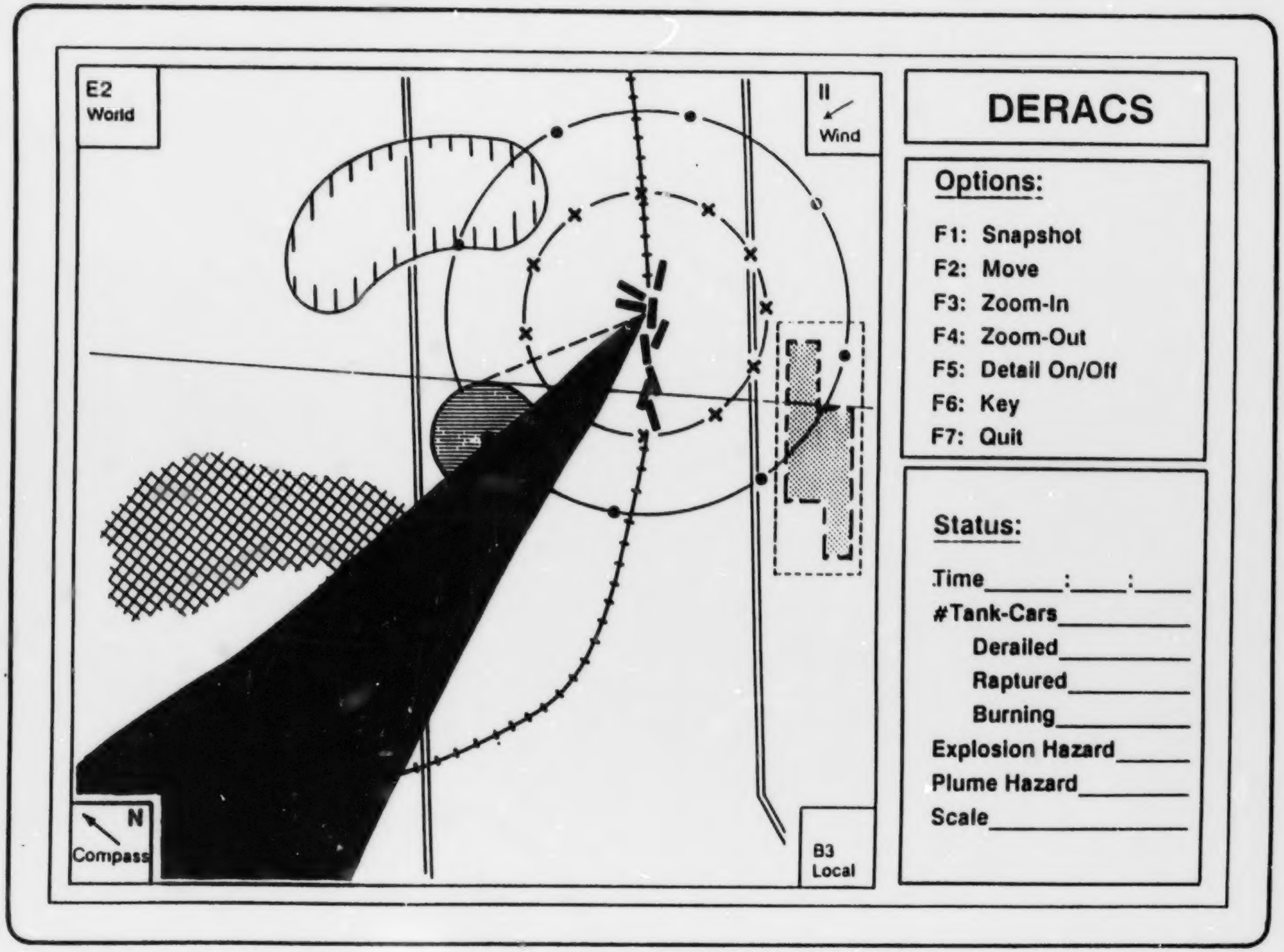

A
B
C
D
E
F
G
H
I
J
1
2
3
4
5
6
7
8
9
10
1 km or 100m
1 Km or 100m
y
x
DERACS
Options:
F1: Snapshot
F2: Move
F3: Zoom-In
F4: Zoom-Out
F5: Detail On/Off
F6: Key
F7: Quit
Enter Grid Location
World
X________(net)
y________(net)
Local
X________(net)
y________(net)

FIGURE 3.4-4 SCREEN IMAGE SHOWING GRID FOR WORLD AND LOCAL LEVELS OF SCALE


E2
World
Wind
N
Compass
B3
Local
DERACS
Options:
F1: Snapshot
F2: Move
F3: Zoom-In
F4: Zoom-Out
F5: Detail On/Off
F6: Key
F7: Quit

Status:

Time______:____:____
#Tank-Cars__________
Derailed__________
Raptured__________
Burning__________
Explosion Hazard______
Plume Hazard________
Scale____________

FIGURE 3.4-5 ACCIDENT VIEW AT LOCAL LEVEL
(NOT TO SCALE)

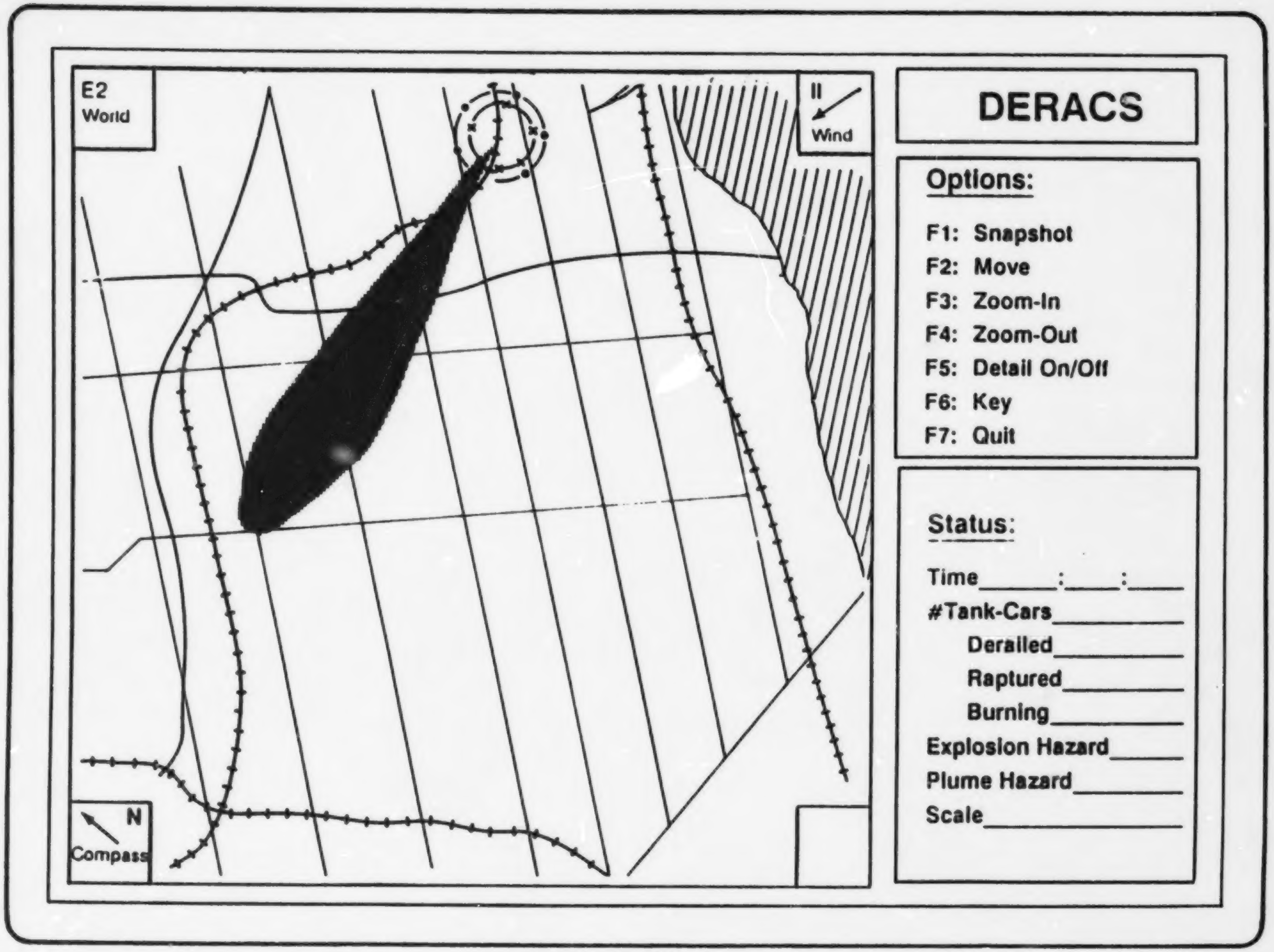

E2
World
Wind
Compass
N
DERACS
Options:
F1: Snapshot
F2: Move
F3: Zoom-In
F4: Zoom-Out
F5: Detail On/Off
F6: Key
F7: Quit
Status:
Time______:____:____
#Tank-Cars__________
Derailed__________
Raptured__________
Burning__________
Explosion Hazard______
Plume Hazard________
Scale_______________

FIGURE 3.4-6 ACCIDENT VIEW AT WORLD LEVEL
(NOT TO SCALE)

4.0 CONCLUSIONS AND RECOMMENDATIONS

Based on the results of the literature search and the preliminary software design, the following conclusions have been made.

1) The various submodels required to develop the accident model are available either in coded or uncoded form.

2) The development required to put these various submodels into the desired format and condition is believed to be within the technical scope of this project.

3) The only submodel identified as requiring significant modification or development is the derailment mechanics submodel and it has been concluded that this submodel should be modified to account for vehicle uncoupling so that tank side coupler impacts can be accounted for. Also, the model should be developed to allow for tank roll so that burning flare impingement on other cars can be accounted for. Although this task is technically feasible within the project, it may require a small adjustment to the project budget.

Based on the above conclusions, it is recommended that the project continue as described in the original project proposal. It is recommended that the derailment mechanics submodel modifications (to allow for uncoupling and roll) be investigated and budgetary and schedule impacts be estimated and presented to the scientific authorities for consideration.

REFERENCES

2.1 Derailment Kinematics

2.1-1 Yang, T.H., Manos, W.P. and Johnstone, B., "Dynamic Analysis of Train Derailments", ASME 72-WA/RT-6, ASME Winter Annual Meeting, November, 1972.

2.1-2 Yang, T.H., et al., "A Study of Continuation of Derailment Behavior: Final Report", Pullman-Standard Research Project No. 39-1833, RPI-AAR Railroad Tank Car Safety Research and Test Project, Reprot RA-08-1-12, February, 1972.

2.1-3 Phillips, E.A. and Role, H., "Effectiveness of Shelf Couplers, Head Shields and Thermal Shields on DOT 112(114) and 105 Tank Cars", RPI-AAR Railroad Tank Car Safety Research and Test Project, Report RA-02-5-51, June, 1985.

2.1-4 Ward, E.d. and Leonard, R.G., "Automatic Parameter Identification Applied to a Railroad Car Dynamic Draft Gear Model", ASME 74-WA/Aut-1, ASME Winter Annual Meeting, November, 1974.

2.2 Derailment Impact Induced Ruptures

2.2-1 C. Orth, Federal Railroad Administration, Washington, D.C., personal communication to P.J. Turyk, DAVIS, June 1987,

2.2-2 R. Ehrinbeck, Transportation Systems Center, Cambridge, MA, personal communication to P.J. Turyk, DAVIS, June 1987.

2.2-3 S.K. Yin, et al, "Theoretical Manual and User's Guide: Longitudinal Vertical Train Action Model", Report No. FRA-ORD-76/278, Federal Railroad Administration, Washington, D.C., April 1980.

2.2-4 D.A. Peters, et al, "Tank Car Head Puncture Mechanisms", Report No. FRA-ORD-76/269, Federal Railroad Administration, Washington, D.C., April 1980.

2.2-5 D.A. Peters, et al, "Non-Destructive Impact Between Railroad Cars: Experimental and Analytical Study", Report No. FRA-ORD-76/247, Federal Railroad Administration, Washington, D.C., January 1977.

2.2-6 K.H. Hohenemser, et al, "Computer Simulation of Tank Car Puncture Mechanisms; Classification Yard Accidents", Report No. FRA-ORD-75-23, Federal Railroad Administration, Washington, DC, February 1975.

2.2-7 J.B. Raidt, et al, "Technical Documentation, Detailed Vertical Train Stability Model", Report No. AAR R-261, Association of American Railroads, Chicago, February 1977.

2.2-8 J.B. Raidt, et al, "Vertical Motions During Railcar Impacts", ASME Report No. 85-WA/RT-10, 1975.

2.2-9 E.A. Phillips, "Final Phase 05 Report on Tank Car Head Study", Report No. AAR-R-140 (or RA-05-1-17), Appendix G., Association of American Railroads, Chicago, 1972.

2.3 Spill/Release Analysis

2.3-1 W.G. England, et al, "Development of a Fuel Spill/Vapour Migration Modelling System", Report No. AFWAL-TR-85-2089, Wright-Patterson AFB, Ohio, December 1985.

2.3-2 W. Stiver, et al, "Evaporation Rate of Spills of Hydrocarbons and Petroleum Mixtures", Environmental Science and Technology, V18, p. 834, 1984.

2.3-3 D. Mackay, et al, "Calculation of the Evaporation Rate of Volatile Liquids", Control of Hazardous Material Spills, Proceedings of 1980 National Conference on Control of Hazardous Material Spills, Louisville, May 1980.

2.3-4 M.F. Fingas, et al, "The Impact of Spilled Chemicals into the Environment", 1986 Hazardous Materials Spills Conference, St. Louis, May 1986.

2.3-5 M.T. Fleischer, "SPILLS: An Evaporation/Air Dispersion Model for Chemical Spills on Land", Control of Hazardous Materials Spills, Proceedings of 1980 National Conference on Control of Hazardous Material Spills, Louisville, May 1980.

2.3-6 F.T. Dodge, et al, "Release Rates of Hazardous Chemicals from a Damaged Cargo Vessel", Control of Hazardous Material Spills, Proceedings of 1980 National Conference on Control of Hazardous Material Spills, Louisville, May 1980.

2.3-7 P. Shaw, et al, "Vapourization of Spills of Hazardous Liquids on Land and Water", Report No. SRD-R-100, Safety and Reliability Directorate, United Kingdom Atomic Energy Authority, May 1978.

2.4 Fire Analysis

2.4-1 L. Schalit, et al, "Development of Analytical Fire Models", Report No. FRA-ORD-75/53, Federal Railroad Administration, Washington, D.C., October 1974.

2.4-2 G. Schneyer, et al, "Development of Analytic Fire Models", Systems, Science and Software Inc., Report No. SSS-R-75-2613, LaJolla, CA, April 1975.

2.4-3 G. Schneyer, Systems Science and Software Inc., LaJolla, CA, Personal Communication to P.J. Turyk, DAVIS, July 1987.

2.4-4 J.L. Rand, et al, "Analysis and Experimental Verification of a Combusting Propane Torch Model", Report No. SWRI-02-5045-002, Southwest Research Institute, San Antonion, TX, August 1982.

2.4-5 A.M. Birk, et al, "Model for the Prediction of Radiant Heat Transfer to a Horizontal Cylinder Engulfed in Flames", ASME Paper No. 82-WA/HT-52, 1982.

2.4-6 M. Tunc, et al, "Incident Radiation from an Engulfing Pool Fire to a Horizontal Cylinder, Part 1", Fire Safety Journal, V8, p. 81, 1984/85.

2.4-7 M. Tunc, et al, "Incident Radiation from a Flare to a Horizontal Cylinder, Part II", Fire Safety Journal, V8, p. 89, 1984/85.

2.4-8 W.P. Crocker, et al, "Quantification of Fire Hazards of Liquid Spills from Tank Trucks", Proceedings of Fourth Technical Seminar on Chemical Spills, Environment Canada Report No. EN40-327/1-1987, p. 77.

2.4-9 D.A. Lihou, et al, "Thermal Radiation from Fireballs", Proc. Conf. Assessment of Major Hazards, Institution of Chemical Engineers Symposium Series, No. 71, Pergamon Press, Oxford, 1982, p. 191.

2.4-10 A.F. Roberts, "The Effects of Conditions Prior to Loss of Containment on Fireball Behaviours", Proc. Conf. Assessment of Major Hazards, Institution of Chemical Engineers Symposium Series No. 71, Pergamon Press, Oxford, 1982, p. 181.

2.4-11 J.S. Klement, "Major Hazard Scenarios with Propane Tank Cars", M.Eng. Project Report, Dept. of Chemical Engineering, University of Toronto, 1986.

2.5 Thermal Rupture

2.5-1 K.W. Graves, "Development of a Computer Program for Modelling the Heat Effects on a Railroad Car", Report No. FRA-ORD-D-75-33, Federal Railroad Administration, Washington, D.C., January 1973.

2.5-2 P.R. Wentzell, "Numerical Modelling of a Tank Car Under Fire Conditions", M.A.Sc. Thesis, Dept. of Mechanical Engineering, University of New Brunswick, Fredericton, N.B., 1985.

2.5-3 A.C.M. Sousa, et al, "Thermal Modelling of LPG Tanks Engulfed in Flames", Heat and Technology (Calore e Technologia), V3, p.1 1985.

2.5-4 A.M. Birk, "Validation and Testing of the Tank Car Thermal Analysis Program", Report No. TP6693E, Transport Canada, Ottawa, Ontario, March 1985.

2.5-5 M.R. Johnson, et al, "Temperatures, Pressures, and Liquid Levels of Tank Cars Engulfed in Fires, Volume 1, Results of Parametric Analyses", Report No. FRA-ORD-84/08.I, Federal Railroad Administration, Washington, D.C., June 1984.

2.5-5 M.R. Johnson, et al, "Temperatures, Pressures, and Liquid Levels of Tank Cars Engulfed in Fires, Volume II, Description of Analytical Procedure", Report No. FRA-ORD-84/08.II, Federal Railroad Administration, Washington, D.C., June 1984.

2.5-6 D.L.M. Hunt, et al, "The Behaviour of Tanks Engulfed in Fire - The Development of a Computer Program", Proc. Conf. Assessment and Control of Major Hazards, Institution of Chemical Engineers Symposium, Series No. 93, Pergamon Press, Oxford, 1985, p. 71.

2.5-7 K. Moodie, et al, "The Fire Engulfment of LPG Storage Tanks", Proc. Conf. Assessment and Control of Major Hazards, Institution of Chemical Engineers Symposium Series No. 93, Pergamon Press, Oxford, 1985, p. 87.

2.5-9 G.V. Benyou, L.T. Cowley, L.M. Small, I. Williams, Fire Engulfment of LPG Tank: HEATUP and Predictive Model, Major Hazards in Transport and Storage of Pressure Liquified Gases, University of New Brunswick, August 1987.

2.6 BLEVE/Explosion Analysis

2.6-1 C.A. McDevitt, et al, "What is a BLEVE?", Proceedings of the Fourth Technical Seminar on Chemical Spills, Report No. EN 40-327/1-1987, p. 137, Environment Canada, Ottawa, Ontario, 1987.

2.6-2 R.C. Reid, Massachusetts Institute of Technology, Cambridge MA, Personal Communication to P.J. Turyk, DAVIS, July 1987.

2.6-3 R.C. Reid, "Possible Mechanisms for Pressurized Liquid Tank Explosions or BLEVEs", Science, V203, p. 1263, 1979.

2.6-4 V.J. Clancey, "The Effects of Explosions", Proc. Conf. Assessment of Major Hazards, Institution of Chemical Engineers Symposium Series No. 71, p. 87, Pergamon Press, Oxford, 1982.

2.6-5 J. MacKenzie, et al, "GASEX1-A General One-Dimensional Code for Gas Cloud Explosion", Report No. SRD-R-251, Safety and Reliability Directorate, United Kingdom Atomic Energy Authority, January 1983.

2.6-6 J. MacKenzie, et al, "A Computational Survey of Factors Influencing the Pressure Fields Produced by Unconfined Vapour Cloud Explosions", Report No. SRD-R-262, Safety and Reliability Directorate, United Kingdom Atomic Energy Authority, March 1983.

2.6-7 D. Martin, "GASEX2: A Two-Dimensional Hydrodynamic Code for Use in Gas Cloud Explosion Calculations", Report No. SRD-R-173, Safety and Reliability Directorate, United Kingdom Atomic Energy Authority, February 1983.

2.6-8 D. Martin, "Validation Calculations for the Code GASEX2", Report No. SRD-R-252, Safety and Reliability Directorate, United Kingdom Atomic Energy Authority, August 1983.

2.6-9 G.F. Kinney, et al., Explosive Shocks in Air, 2nd Ed., Springer-Verlag, New York, 1985.

2.6-10 W.E. Baker et al, Explosion Hazards and Evaluations, Elsevier, New York, 1983.

2.6-11 Leuschen, A., Birk, A.M., "Computational Prediction of Air Blast Loads on Simple Structures", BSc. Thesis, Engineering, Queen's University, Kingston Ontario Canada, March 1987.

2.7 Plume Dispersion

2.7-1 S.T. Chan, et al, "FEM3: A Finite Element Model for the Simulation of Heavy Gas Dispersion and Incompressible Flow", Report No. UCRL-53397, Lawrence Livermore National Laboratory, February 1983.

2.7-2 S.T. Chan, et al, "Numerical Simulations of Atmospheric Releases of Heavy Gases over Variable Terrain", Report No. UCRL-87256, Lawrence Livermore National Laboratory, August 1982.

2.7-3 S.T. Chan, et al, "Recent Progress in Modelling the Atmospheric Dispersion of Heavy Gases", Report No. UCRL-88495, Lawrence Livermore National Laboratory, August 1983.

2.7-4 D.C. Morgan, et al, "Phenomenology and Modelling of Heavy Gas Dispersion", 1984 Hazardous Material Spills Conference, Nashville, April 1984.

2.7-5 W.J. Hogan, et al, "Predicting the Hazards from Large Spills of Liquefied Gaseous Fuels", 1982 Hazardous Material Spills Conference, Milwaukee, April 1982.

2.7-6 D.L. Morgan, et al, "SLAB: A Time Dependent Computer Model for the Dispersion of Heavy Gases Released in the Atmosphere", Report No. UCRL-53383, Lawrence Livermore National Laboratory, January 1983.

2.7-7 D.L. Morgan, et al, "Simulations and Parameter Variation Studies of Heavy Gas Dispersion Using the SLAB Model", Report No. UCRL-88516, Revision 1, Lawrence Livermore National Laboratory, August 1983.

2.7-8 D.L. Morgan, et al, "Simulations and Studies of Heavy Gas Dispersion Using the SLAB Model", Report No. UCRL-88009, Lawrence Livermore National Laboratory, November 1982.

2.7-9 G. Tsang, "Concentration of Effluents in a Plume as Predicted by Model and Observed in Field", Report 69-7, MIT, Cambridge MA, August 1969.

2.7-10 R.C. Koch, et al, "Validation and Sensitivity Analysis of the Gaussian Plume Multiple Source Urban Diffusion Model", Report No. EF-60, Geomet Inc., Rockville MD, November 1971.

2.7-11 W.B. Petersen, "Estimating Concentrations Downwind from an Instantaneous Puff Release", Report No. EPA-600/3-82-078, Environmental Sciences Research Lab, Research Triangle Park, NC, August 1982.

2.7-12 SAFEMODE Computer Code Demonstration Manual, Technology and Management Systems Inc., Cambridge, MA, 1987.

2.7-13 P.K. Raj, "Dispersion of Hazardous Reactive Chemical Vapours in the Atmosphere", 1986 Hazardous Material Spills Conference, St. Louis, May 1986.

2.7-14 P.K. Raj, "Summary of Heavy Gas Spills Modelling Research", Proceedings of Heavy Gas (LNG/LPG Workshop), Toronto, 1985.

2.7-15 E. Alp, "COBRA: An LNG Model", Proceedings of Heavy Gas (LNG/LPG) Workshop, Toronto, 1985.

2.7-16 T.L. DeVaan, "User's Manual for Camoe Air Model ALOHA", HAZMAT Branch, NOAA, Seattle, 1985.

2.7-17 F. Pasquill, Atmospheric Diffusion, John Wiley & Sons, New York, 1974.

2.8 Other Resources

2.8-1 T.S. Glickman, "Data Requirements for Hazardous Materials Transportation Risk Analysis", 1984 Hazardous Material Spills Conference, Nasville, April 1984.

2.8-2 D.H. Napier, "Review of Important Technical Considerations in Heavy Gas Risk Assessment", Proceedings of Heavy Gas (LNG/LPG) Workshop, Toronto 1985.

2.8-3 G.R. Colonna, "Hazard Assessment Computer System (HACS) Models Reviews and Validation, 1982 Hazardous Material Spills Conference, Milwaukee, 1982.

2.8-4 Atmospheric Dispersion of Hazardous/Toxic Materials from Transport Accidents, W.F. Dabberdt, Ed., Elsevier, Amsterdam, 1984.

2.8-5 Atmospheric Dispersion of Heavy Gases And Small Particles, G. Ooms and H. Tennekes, Eds., Springer-Verlay, Berlin, 1984.

APPENDIX B

PROGRAM LISTING

```
      PROGRAM TT
      INTEGER          NBLUE,GREEN,LBLUE,RED,VIOLET,BROWN,
     1                 WHITE,GRAY,BBLUE,BGREEN,BBBLUE,BRED,
     1                 BVIOLT,YELLOW,BWHITE
      COMMON/COLORS/   NBLUE,GREEN,LBLUE,RED,VIOLET,BROWN,
     1                 WHITE,GRAY,BBLUE,BGREEN,BBBLUE,BRED,
     1                 BVIOLT,YELLOW,BWHITE
      COMMON/ARRAS/    INTARR(100,10),REARR(100,100,2),KOS
      COMMON/WORLDC/   XB,YB,XT,YT,ILEVEL
      COMMON/DERDAT/   RCARS(40,13),NCARS,UWIND,UANGL,PATM,
     1                 TATM,UTRAIN,RCURV,ICARDR,ITRUCK,IAXLE,
     1                 CENTER,D1X,D1Y,D2X,D2Y,
     1                 USTR,UEND,DX,DY,IDTRK,ISPEED,CENX,CENY,
     1                 NUMTNK,DERAIL,DERTYP,ICAR,ATMOSP
      COMMON/RESULT1/  IBLAST(8),BLAST(8,7),IFLARE(8),FLAR(8,3),
     1                 IPUFF(8),PUF(8,3),ISP(8),SP(8,6),
     1                 IPLUM(8),PLM(8,2),PLMM(300,2),
     1                 IFIRE(8),FIRE(8,4),RUPDAT(8,5),IPLM(8),IPUF(8)
      COMMON/GLBTM/    IGLBTM
      COMMON/DET/      IDETAL(7)
      COMMON/MODE/     IMODE
      COMMON/BLK/      IBLACK
      COMMON/TRK/      IWIDTH
      COMMON/WINDS/    WSPEED
      COMMON/TRKK/     ITRACK
      COMMON/DIRC/     DIRECT
      COMMON/ROTATE/   BETA
      COMMON/RUPT/      RPT(8)
      COMMON/TRANS/    THETA


      WSPEED = 0.
      DIRECT = 1.0
      THETA = 0.
      BETA = 0.
      ITRACK = 1
      IWIDTH = 1
      IBLACK = 0
      ILEVEL = 1
      IMODE = 1
      XB = 0.
      YB = 0.
      XT = 10000.
      YT = 10000.

      CALL SETC()
      CALL DRW()
10000 WRITE(*,*)' 11;10HPROGRAM   1mEXIT0m'
      END

***************************************************************************
***************************************************************************
```

```
******************************************************************************************
******************************************************************************************
      SUBROUTINE DRW
$STORAGE:2
      INTEGER          SWITCH
      REAL             XR,YR
      LOGICAL          CLICK,EXITT

      INTEGER          NBLUE,GREEN,LBLUE,RED,VIOLET,BROWN,
     1                 WHITE,GRAY,BBLUE,BGREEN,BBBLUE,BRED,
     1                 BVIOLT,YELLOW,BWHITE
      COMMON/COLORS/   NBLUE,GREEN,LBLUE,RED,VIOLET,BROWN,
     1                 WHITE,GRAY,BBLUE,BGREEN,BBBLUE,BRED,
     1                 BVIOLT,YELLOW,BWHITE
      COMMON/ARRAS/    INTARR(100,10),REARR(100,100,2),KOS
      COMMON/WORLDC/   XB,YB,XT,YT,ILEVEL
      COMMON/DERDAT/   RCARS(40,13),NCARS,UWIND,UANGL,PATM,
     1                 TATM,UTRAIN,RCURV,ICARDR,ITRUCK,IAXLE,
     1                 CENTER,D1X,D1Y,D2X,D2Y,
     1                 USTR,UEND,DX,DY,IDTRK,ISPEED,CENX,CENY,
     1                 NUMTNK,DERAIL,DERTYP,ICAR,ATMOSP
      COMMON/POS1/         XX,YY
      COMMON/WINDS/    WSPEED
      COMMON/TRKK/     ITRACK

      WRITE(*,*)' 2J'

      CALL SETDEV('/HALOIBME.DEV/')
      CALL INITGR(4)
      CALL SETIEE(1)
      CALL SETASP(.75)
      CALL SETLOC(2,2)
      CALL SETLAT(0)
      CALL SETVIE(0.,0.,.746875,.85,4,2)

      CALL DRICON()

*************************************************
      KOS = 3
      XX = 50.
      YY = 500.
      CALL SETLOC(2,2)
      CALL SETLAT(0)
      CALL SETLNW(1)
      CALL SETLNW(1)
      CALL SETLNS(1)
      CALL SETLNS(1)
      CALL SETVIE(0.,0.,.746875,.85,0,GREEN)
      CALL UPDATE()
      CALL RESETW()

40    CALL READLO(XR,YR,SWITCH)
```

```
      CALL MOVHCA(XR,YR)

      IF(XR .LT. XT) THEN
         CONTINUE
      ELSE
50       CALL WHICH(ICNUM,CLICK)
         IF (CLICK) THEN
             IF (ICNUM .EQ. 1 ) THEN
                CALL SUBZOM(2)
                CALL DRICON()
             ENDIF
             IF (ICNUM .EQ. 2 ) CALL SUBRDR()
             IF (ICNUM .EQ. 3 ) CALL KEY()
             IF (ICNUM .EQ. 4 ) THEN
                CALL SUBZOM(1)
                CALL DRICON()
             ENDIF
             IF (ICNUM .EQ. 5 ) CALL UNDO()
             IF (ICNUM .EQ. 6) CALL EXITP(EXITT)
             IF (ICNUM .EQ. 7) CALL DRAW1()
             IF (ICNUM .EQ. 8) CALL TRAK()
             IF (ICNUM .EQ. 9) THEN
                CALL SUBATC()
                CALL DRICON()
             ENDIF
             IF (ICNUM .EQ. 10 ) THEN
                CALL SUBDER()
                CALL SETSEG(10)
                CALL SETDEV('/HALOIBME.DEV/')
                CALL INITGR(4)
                CALL SETIEE(1)
                CALL SETASP(.75)
                CALL DRICON()
                WRITE(*,*)' 22;0H'
                CALL SUBRDR()
             ENDIF
             IF (ICNUM .EQ. 11) CALL DRAW11(7)
             IF (ICNUM .EQ. 12) CALL DRAW11(8)
             IF (ICNUM .EQ. 13) CALL DRAW11(9)
             IF (ICNUM .EQ. 14) CALL DRAW10()
             IF (ICNUM .EQ. 15) CALL DRAW11(12)
             IF (ICNUM .EQ. 16) CALL DRAW11(13)
             IF (ICNUM .EQ. 17) CALL DRAW11(11)
             IF (ICNUM .EQ. 18) CALL DRAW11(14)
             IF (ICNUM .EQ. 19) CALL TDEMO()
             IF (EXITT) THEN
                GOTO 10000
             ENDIF
             CALL UPDATE()
         ENDIF
         CALL READLO(XR,YR,SWITCH)
         IF (SWITCH .EQ. 2 .OR. SWITCH .EQ. 130) THEN
```

```
              CALL WAIT2()
              CALL RESETW()
              GOTO 40
           ENDIF
           GOTO 50
        ENDIF
        GOTO 40
10000   END

************************************************************************
************************************************************************

        SUBROUTINE TDEMO()
        CHARACTER    JUNK

        CALL SETVIE(.76,.5,.98,.80,0,1)
        CALL SETWOR(0.,16.,10.,0.)
        CALL SETTEX(1,1,0,0)
        CALL SETTCL(12,8)
        CALL MOVTCA(1.,2.)
        CALL TEXT('/CURRENT PICTURE/')
        CALL MOVTCA(1.,4.)
        CALL TEXT('/WILL BE ERASED!/')
        CALL MOVTCA(1.,10.)
        CALL TEXT('/<C> TO CONTINUE/')
        CALL DELTCU()
        CALL PROMPT()
        READ(*,'(A)',END=20,ERR=20)JUNK
20      CALL CPROM()
        CALL DRICON()
        IF (JUNK .EQ. 'C' .OR. JUNK .EQ. 'c') THEN
           CALL RCRS(0)
           CALL SUBRDR()
        ENDIF
10000   END


************************************************************************
************************************************************************

        SUBROUTINE UNDO()

        COMMON/ARRAS/    INTARR(100,10),REARR(100,100,2),KOS



        IF (KOS .GT. 3) THEN
           KOS = KOS - 1
           INTARR(KOS,1) = 0
           INTARR(KOS,2) = 0
           DO 100 I = 1,100
              REARR(KOS,I,1) = 0.
              REARR(KOS,I,2) = 0.
100         CONTINUE

            CALL SUBRDR()
         ENDIF
         END

******************************************************************************
******************************************************************************

         SUBROUTINE DRICON
         INTEGER           NBLUE,GREEN,LBLUE,RED,VIOLET,BROWN,
     1                     WHITE,GRAY,BBLUE,BGREEN,BBBLUE,BRED,
     1                     BVIOLT,YELLOW,BWHITE
         COMMON/COLORS/    NBLUE,GREEN,LBLUE,RED,VIOLET,BROWN,
     1                     WHITE,GRAY,BBLUE,BGREEN,BBBLUE,BRED,
     1                     BVIOLT,YELLOW,BWHITE
         COMMON/BLK/       IBLACK


         CALL SETLNS(1)
         CALL SETLNS(1)
         CALL SETLNW(1)
         CALL SETLNW(1)
         CALL SETVIE(.746875,0.,1.,.15,0,GRAY)
         CALL SETWOR(0.,10.,10.,0.)
         CALL SETTEX(2,2,0,0)
         CALL SETTCL(NBLUE,GRAY)
         CALL MOVTCA(2.,7.)
         CALL TEXT('/DERACS/')

         CALL SETVIE(.746875,.15,1.,.85,0,GRAY)
         CALL SETWOR(0.,1000.,90.,0.)

************   ICON #1
         CALL SETCOL(0)
         CALL SETCOL(0)
         CALL SETHAT(0)
         CALL SETHAT(0)
         CALL SETTEX(1,1,0,0)
         CALL SETTCL(BRED,IBLACK)

         CALL DELTCU()
*********** ICON #1
         CALL BAR(2.,50.,28.,100.)
         CALL MOVTCA(10.,90.)
         CALL SETTCL(BRED,IBLACK)
         CALL TEXT('/ZIN/')
         CALL DELTCU()

*********** ICON #2
         CALL BAR(32.,50.,58.,100.)
```

```
      CALL MOVTCA(38.,90.)
      CALL SETTCL(BRED,IBLACK)
      CALL TEXT('/VIEW/')
      CALL DELTCU()

*********** ICON #3
      CALL BAR(62.,50.,88.,100.)
      CALL MOVTCA(68.,90.)
      CALL SETTCL(BRED,IBLACK)
      CALL TEXT('/KEY/')
      CALL DELTCU()


*********** ICON #4
      CALL BAR(2.,150.,28.,200.)
      CALL MOVTCA(8.,190.)
      CALL SETTCL(BRED,IBLACK)
      CALL TEXT('/ZOUT/')
      CALL DELTCU()

*********** ICON #5
      CALL BAR(32.,150.,58.,200.)
      CALL MOVTCA(38.,190.)
      CALL SETTCL(BRED,IBLACK)
      CALL TEXT('/UNDO/')
      CALL DELTCU()

*********** ICON #6
      CALL BAR(62.,150.,88.,200.)
      CALL MOVTCA(66.,190.)
      CALL SETTCL(BRED,IBLACK)
      CALL TEXT('/EXIT/')
      CALL DELTCU()


*********** ICON #7
      CALL BAR(2.,300.,28.,350.)
      CALL MOVTCA(5.,340.)
      CALL SETTCL(BRED,IBLACK)
      CALL TEXT('/TRAIN/')
      CALL DELTCU()

*********** ICON #8
      CALL BAR(32.,300.,58.,350.)
      CALL MOVTCA(36.,340.)
      CALL SETTCL(BRED,IBLACK)
      CALL TEXT('/TRACK/')
      CALL DELTCU()

*********** ICON #9
      CALL BAR(62.,300.,88.,350.)
      CALL MOVTCA(70.,340.)
```

```
        CALL SETTCL(BRED,IBLACK)
        CALL TEXT('/ATM/')
        CALL DELTCU()


        CALL BAR(10.,400.,80.,450.)
        CALL MOVTCA(13.,440.)
        CALL SETTCL(BRED,IBLACK)
        CALL TEXT('/PUT CAR OPTIONS/')
        CALL DELTCU()



************  ICON #
        CALL SETCOL(BROWN)
        CALL SETLNS(1)
        CALL SETLNW(7)
        CALL MOVABS(5.,575.)
        CALL LNREL(23.,0.)

************  ICON #
        CALL SETLNW(5)
        CALL MOVABS(33.,575.)
        CALL LNREL(23.,0.)

************  ICON #
        CALL SETLNW(3)
        CALL MOVABS(60.,575.)
        CALL LNREL(23.,0.)




**********  ICON #
        CALL SETLNS(1)
        CALL SETHAT(1)
        CALL SETCOL(BBLUE)
        CALL BAR(5.,650.,23.,700.)

**********   ICON #
        CALL SETLNS(1)
        CALL SETLNW(1)
        CALL SETCOL(NBLUE)
        CALL SETHAT(4)
        CALL BOX(33.,650.,53.,700.)
        CALL MOVABS(35.,655.)
        CALL FLOOD(3)

**********   ICON #
        CALL SETCOL(BGREEN)
        CALL SETHAT(3)
        CALL BOX(60.,650.,83.,700.)
        CALL MOVABS(65.,655.)
        CALL FLOOD(BGREEN)




***********  ICON #
        CALL SETCOL(LBLUE)
        CALL SETCOL(LBLUE)
        CALL SETLNS(1)
        CALL SETLNW(1)
        CALL MOVABS(5.,775.)
        CALL LNREL(25.,0.)


***********  ICON #
        CALL SETCOL(VIOLET)
        CALL SETCOL(VIOLET)
        CALL SETLNW(1)
        CALL MOVABS(35.,775.)
        CALL LNREL(25.,0.)


***********  ICON #16
        CALL SETHAT(1)
        CALL SETCOL(BGREEN)
        CALL BAR(20.,850.,70.,900.)
        CALL SETTEX(1,1,0,0)
        CALL SETTCL(BRED,RED)
        CALL MOVTCA(36.,890.)
        CALL TEXT('/DEMO/')
        CALL DELTCU()

10000   END

************************************************************************
************************************************************************

        SUBROUTINE  WHICH(ICNUM,CLICK)
        INTEGER       ICNUM
        LOGICAL       CLICK
        INTEGER       SWITCH
        REAL          X,Y
        COMMON/POS1/         XX,YY

        CLICK = .FALSE.
        CALL LEFT()
        CALL INITHC(30.,10.,13)
        CALL ORGLOC(XX,YY)
```

```
40    CALL READLO(X,Y,SWITCH)
      IF (X .GE. 90.) THEN
         X = 88.
      ELSEIF (X .LE. 0.) THEN
         X = 2.
      ELSEIF (Y .GE. 1000.) THEN
         Y = 998.
      ELSEIF (Y .LE. 0.) THEN
         Y = 2.
      ENDIF
      CALL MOVHCA(X,Y)
      CALL ORGLOC(X,Y)
      IF (SWITCH .EQ. 4 .OR. SWITCH .EQ. 132) THEN
            XX = X
            YY = Y
            CALL FICON(X,Y,ICNUM)
50          IF (ICNUM .NE. 0) CLICK = .TRUE.
            CALL WAIT4()
            GOTO 10000
      ELSEIF (SWITCH .EQ. 2 .OR. SWITCH .EQ. 130) THEN
           GOTO 10000
      ENDIF
      GOTO 40
10000 END

************************************************************************
************************************************************************

      SUBROUTINE FICON(X,Y,ICNUM)
      REAL       X,Y
      INTEGER    ICNUM



      ICNUM = 0
      IF (Y .LE. 200.) THEN
         IF (Y .LE. 100.) THEN
            ICNUM = 1
         ELSEIF (Y .LE. 200. .AND. Y .GT. 150.) THEN
            ICNUM = 4
         ELSE
            GOTO 10
         ENDIF
         IF (X .GE. 2. .AND. X .LE. 28.) THEN
            ICNUM = ICNUM + 0.
         ELSEIF (X .GE. 32. .AND. X .LE. 58.) THEN
            ICNUM = ICNUM + 1.
         ELSEIF (X .GE. 62. .AND. X .LE. 88.) THEN
            ICNUM = ICNUM + 2.
         ELSE
            CONTINUE
         ENDIF
```

```
10          CONTINUE
         ELSEIF (Y .GT. 300. .AND. Y .LE. 450.) THEN
            IF (Y .GE. 300. .AND. Y .LE. 350.) THEN
               ICNUM = 7
            ELSEIF (Y .GE. 400. .AND. Y .LT. 450.) THEN
               ICNUM = 10
            ELSE
               GOTO 20
            ENDIF
            IF (ICNUM .NE. 10 .AND. X .GE. 2. .AND. X .LE. 28.) THEN
               ICNUM = ICNUM + 0
            ELSEIF (ICNUM .NE. 10 .AND. X .GE. 32. .AND. X .LE. 58.)THEN
               ICNUM = ICNUM + 1
            ELSEIF (ICNUM .NE. 10 .AND. X .GE. 62. .AND. X .LE. 88.)THEN
               ICNUM = ICNUM + 2
            ELSE
               CONTINUE
            ENDIF
20          CONTINUE
         ELSEIF (Y .GE. 550. .AND. Y .LE. 800.) THEN
            IF (Y .GE. 550. .AND. Y .LE. 600.)  THEN
               ICNUM = 11
            ELSEIF (Y .GE. 650. .AND. Y .LE. 700.) THEN
               ICNUM = 14
            ELSEIF (Y .GE. 750. .AND. Y .LE. 800.) THEN
               ICNUM = 17
            ELSE
               GOTO 30
            ENDIF
            IF (X .GE. 5. .AND. X .LE. 28.) THEN
               ICNUM = ICNUM + 0
            ELSEIF (X .GE. 33. .AND. X .LE. 53.) THEN
               ICNUM = ICNUM + 1
            ELSEIF (X .GE. 60. .AND. X .LE. 83.) THEN
               ICNUM = ICNUM + 2
            ELSE
               CONTINUE
            ENDIF
30          CONTINUE
         ELSEIF (Y .GE. 850. .AND. Y .LE. 900.) THEN
            ICNUM = 19
         ENDIF
10000    END

***********************************************************************
***********************************************************************

         SUBROUTINE CONTUR
         INTEGER          KEYIN,SWITCH,ELEVA
         REAL             X,Y,X1,Y1
         REAL             WX1,WY1,WX2,WY2
         LOGICAL          DBG
```

```
      INTEGER          NBLUE,GREEN,LBLUE,RED,VIOLET,BROWN,
     1                 WHITE,GRAY,BBLUE,BGREEN,BBBLUE,BRED,
     1                 BVIOLT,YELLOW,BWHITE
      COMMON/COLORS/   NBLUE,GREEN,LBLUE,RED,VIOLET,BROWN,
     1                 WHITE,GRAY,BBLUE,BGREEN,BBBLUE,BRED,
     1                 BVIOLT,YELLOW,BWHITE
      COMMON/ARRAS/    INTARR(100,10),REARR(100,100,2),KOS


      CALL DWIN()
      CALL SETTEX(1,1,0,0)
      CALL SETTCL(12,2)
      CALL MOVTCA(1.,2.)
      CALL TEXT('/ENTER/')
      CALL MOVTCA(1.,4.)
      CALL TEXT('/ELEVATION/')
      CALL MOVTCA(1.,6.)
      CALL TEXT('/OF CONTOUR:/')
      CALL MOVTCA(1.,10.)
      CALL TEXT('/...Meters.../')
      CALL DELTCU()
20    WRITE(*,'(A\)')' 23;60H                      '
      WRITE(*,'(A\)')' 23;63H'
      WRITE(*,'(A\)')' -> 0'
      WRITE(*,'(A\)')' 23;63H'
      WRITE(*,'(A\)')' -> '
      READ(*,'(I10)',END=20,ERR=20)ELEVA
      IF (ELEVA .GT. 40000 .OR. ELEVA .LT. -20000) GOTO 20
      INTARR(KOS,3) = INT(ELEVA)
      CALL CPROM()
      CALL DRICON()
10000 END
```

```
************************************************************************
************************************************************************


        SUBROUTINE DWIN()
$STORAGE:2
        CALL SETVIE(.75,.5,.99,.80,0,1)
        CALL SETWOR(0.,12.,10.,0.)
10000   END


************************************************************************
************************************************************************

        SUBROUTINE INIPST()

        INTEGER          NBLUE,GREEN,LBLUE,RED,VIOLET,BROWN,
     1                  WHITE,GRAY,BBLUE,BGREEN,BBBLUE,BRED,
     1                  BVIOLT,YELLOW,BWHITE
        LOGICAL          OUTFLG
        COMMON/COLORS/   NBLUE,GREEN,LBLUE,RED,VIOLET,BROWN,
     1                  WHITE,GRAY,BBLUE,BGREEN,BBBLUE,BRED,
     1                  BVIOLT,YELLOW,BWHITE
        COMMON/WORLDC/   XB,YB,XT,YT,ILEVEL
        COMMON/DET/      IDETAL(7)
        COMMON/MODE/     IMODE
        COMMON/OUTFL/    OUTFLG
        COMMON/WINDS/   WSPEED

        ILEVEL = 1

        XB = 0.
        YB = 0.
        XT = 10000.
        YT = 10000.

        CALL SETASP(.75)

        DO 111 I = 1,7
           IDETAL(I) = 1
111     CONTINUE

        CALL SETC()

10000   END


************************************************************************
************************************************************************

        SUBROUTINE UPDATE()
```

```
$STORAGE:2
      INTEGER            IOTHER
      COMMON/WORLDC/     XB,YB,XT,YT,ILEVEL
      COMMON/DERDAT/
     1                   RCARS(40,13),NCARS,UWIND,UANGL,PATM,
     1                   TATM,UTRAIN,RCURV,ICARDR,ITRUCK,IAXLE,
     1                   CENTER,D1X,D1Y,D2X,D2Y,
     1                   USTR,UEND,DX,DY,IDTRK,ISPEED,CENX,CENY,
     1                   NUMTNK,DERAIL,DERTYP,ICAR,ATMOSP

      CALL SETLEV()
      IOTHER = NCARS - NUMTNK
      WRITE(*,'(A,I1,A,I2,A,I2)')' 23;0HZOOM: ',ILEVEL,
     1          '  TANK CARS: ',NUMTNK,'  OTHER CARS: ',IOTHER
      WRITE(*,'(A,4(F6.0,A),A)')' 24;0HWORLD: ',
     1     XB,',',YB,',',XT,',',YT,' 22;0H'
      END

************************************************************************
************************************************************************

      SUBROUTINE SETHT()
      INTEGER    ARR(18)
      DATA ARR/4,4,0,2,2,0,2,0,0,2,2,0,0,2,0,2,2,0/
      CALL DEFHAT(6,ARR(1))
10000 END

************************************************************************
************************************************************************

      SUBROUTINE SUBDIS(X1,Y1,X2,Y2,DIST,TH)

      REAL        X1,Y1,X2,Y2,DIST,TH

      X2 = DIST * COS(TH) + X1
      Y2 = DIST * SIN(TH) + Y1
10000 END

************************************************************************
************************************************************************

      SUBROUTINE FIXLN()

      CALL SETLNS(1)
      CALL SETLNS(1)
      CALL SETCOL(4)
      CALL SETCOL(4)
      CALL SETLNW(1)
      CALL SETLNW(1)
      CALL SETHAT(0)
      CALL SETHAT(0)
10000 END
```

```
************************************************************************
************************************************************************

      SUBROUTINE SUBROT(X1,Y1,X2,Y2,TH)
      REAL        X1,Y1,X2,Y2,TH

      XM = XR - X1
      TH = TH / 57.2957795
      A = COS(TH)
      B = SIN(TH)
      X2 = A*XM
      Y2 = B*XM
      X2 = X2 + X1
      Y2 = Y2 + Y1
10000  END

************************************************************************
************************************************************************

        SUBROUTINE RESETW()

        REAL                  XM,YM,XX,YY
        INTEGER          NBLUE,GREEN,LBLUE,RED,VIOLET,BROWN,
     1                   WHITE,GRAY,BBLUE,BGREEN,BBBLUE,BRED,
     1                   BVIOLT,YELLOW,BWHITE
        COMMON/COLORS/   NBLUE,GREEN,LBLUE,RED,VIOLET,BROWN,
     1                   WHITE,GRAY,BBLUE,BGREEN,BBBLUE,BRED,
     1                   BVIOLT,YELLOW,BWHITE
        COMMON/WORLDC/         XB,YB,XT,YT,ILEVEL

        CALL SETVIE(0.,0.,.0001,.0001,0,GREEN)
        CALL SETVIE(0.,0.,.746875,.85,-1,-1)
        CALL SETWOR(XB,YT,XT,YB)
        CALL SETCLI(XB,YB,XT,YT)
        XM = (XT - XB)/2. + XB
        YM = (YT - YB)/2. + YB
        XX = (XT - XB)
        YY = (YT - YB)
        CALL ORGLOC(XM,YM)
        CALL INITHC(XX/40.,YY/40.,0)
10000   END

************************************************************************
************************************************************************

        SUBROUTINE SETLEV()

        COMMON/WORLDC/         XB,YB,XT,YT,ILEVEL

        IF ((XT-XB) .GT. 9000.) THEN
           ILEVEL = 1
        ELSEIF ((XT-XB) .GT. 900) THEN
           ILEVEL = 2
        ELSE
           ILEVEL = 3
        ENDIF

10000   END
************************************************************************
************************************************************************

        SUBROUTINE PROMPT()
        WRITE(*,'(A\)')' 23;60H                '
        WRITE(*,'(A\)')' 23;63H'
        WRITE(*,'(A\)')' ->'
10000   END
************************************************************************
************************************************************************

        SUBROUTINE LEFT()


        CALL SETVIE(.746875,.15,1.,.85,-1,-1)
        CALL SETWOR(0.,1000.,90.,0.)
10000   END

************************************************************************
************************************************************************

        SUBROUTINE CPROM()
        WRITE(*,'(A\)')' 23;60H                   '
10000   END


************************************************************************
************************************************************************

        SUBROUTINE WAIT4
        INTEGER       SWITCH
        REAL          X,Y
10      CALL READLO(X,Y,SWITCH)
        IF (SWITCH .EQ. 4 .OR. SWITCH .EQ. 132) THEN
            CONTINUE
        ELSE
            GOTO 10000
        ENDIF
        GOTO 10
10000   END

************************************************************************
************************************************************************

        SUBROUTINE WAIT2
        INTEGER        SWITCH
        REAL           X,Y
10      CALL READLO(X,Y,SWITCH)
        IF (SWITCH .EQ. 2 .OR. SWITCH .EQ. 130) THEN
            CONTINUE
        ELSE
            GOTO 10000
        ENDIF
        GOTO 10
10000   END

************************************************************************
************************************************************************

        SUBROUTINE STALL

        DO 10 I = 1,1
10      CONTINUE

10000   END

************************************************************************
************************************************************************

        SUBROUTINE SETC()

        INTEGER            NBLUE,GREEN,LBLUE,RED,VIOLET,BROWN,
     1                     WHITE,GRAY,BBLUE,BGREEN,BBBLUE,BRED,
     1                     BVIOLT,YELLOW,BWHITE
        COMMON/COLORS/     NBLUE,GREEN,LBLUE,RED,VIOLET,BROWN,
     1                     WHITE,GRAY,BBLUE,BGREEN,BBBLUE,BRED,
     1                     BVIOLT,YELLOW,BWHITE
        COMMON/BLK/        IBLACK


        IBLACK     =  0
        NBLUE      =  1
        GREEN      =  2
        LBLUE      =  3
        RED        =  4
        VIOLET     =  5
        BROWN      =  6
        WHITE      =  7
        GRAY       =  8
        BBLUE      =  9
        BGREEN     =  10
        BBBLUE     =  11
        BRED       =  12
        BVIOLT     =  13
        YELLOW     =  14
        BWHITE     =  15
10000   END
```

```
************************************************************************
************************************************************************

        SUBROUTINE DISXY(X,Y)
        REAL             X,Y

        COMMON/WORLDC/   XB,YB,XT,YT,ILEVEL
        IF     (X .LT. XB) THEN
            X = XB
        ELSEIF (X .GT. XT) THEN
            X = XT
        ELSEIF (Y .LT. YB) THEN
            Y = YB
        ELSEIF (Y .GT. YT) THEN
            Y = YT
        ENDIF
        CALL ORGLOC(X,Y)
        WRITE(*,'(A,F6.0,A,F6.0,A)')' 23;50H',X,',',Y,' 22;0H'
        END

************************************************************************
************************************************************************

        SUBROUTINE CLRXY()
        WRITE(*,'(A\)')' 23;50H                    '
10000   END

************************************************************************
************************************************************************
```

```
        SUBROUTINE DRAW10
$STORAGE:2
        INTEGER          SWITCH
        REAL             X1,Y1,X2,Y2
        REAL             WX1,WY1,WX2,WY2
        INTEGER          NBLUE,GREEN,LBLUE,RED,VIOLET,BROWN,
     1                   WHITE,GRAY,BBLUE,BGREEN,BBBLUE,BRED,
     1                   BVIOLT,YELLOW,BWHITE
        COMMON/COLORS/   NBLUE,GREEN,LBLUE,RED,VIOLET,BROWN,
     1                   WHITE,GRAY,BBLUE,BGREEN,BBBLUE,BRED,
     1                   BVIOLT,YELLOW,BWHITE
        COMMON/ARRAS/    INTARR(100,10),REARR(100,100,2),KOS
        COMMON/WORLD/    WX1,WY1,WX2,WY2



        CALL WAIT4()
        INTARR(KOS,1) = 10
        INTARR(KOS,2) = 1
        CALL RESETW()

30      CALL READLO(X1,Y1,SWITCH)
        CALL DISXY(X1,Y1)
        CALL MOVHCA(X1,Y1)
        CALL MOVABS(X1,Y1)
        IF (SWITCH .EQ. 4 .OR. SWITCH .EQ. 132) THEN
           CALL CLRXY()
           CALL WAIT4()
           CALL SETXOR(1)
           CALL FIXLN()
           REARR(KOS,1,1) = X1
           REARR(KOS,1,2) = Y1
           CALL DELHCU()
50         CALL READLO(X2,Y2,SWITCH)
           CALL DISXY(X2,Y2)
           CALL RBOX(X1,Y1,X2,Y2)
           IF (SWITCH .EQ. 4 .OR. SWITCH .EQ. 132) THEN
              CALL CLRXY()
              CALL DELBOX()
              CALL WAIT4()
              CALL SETXOR(0)
              CALL SETCOL(BBLUE)
              REARR(KOS,2,1) = X2
              REARR(KOS,2,2) = Y2
              INTARR(KOS,2) = 2
              CALL SUBBOX()
              KOS = KOS + 1
              GOTO 10000
           ELSEIF( SWITCH .EQ. 2 .OR. SWITCH .EQ. 130) THEN
                  CALL DELBOX()
                  CALL WAIT2()
                  GOTO 10000
```

```
          ENDIF
          GOTO 50
       ELSEIF (SWITCH .EQ. 2 .OR. SWITCH .EQ. 130) THEN
          CALL WAIT2()
          GOTO 10000
       ENDIF
       GOTO 30
10000  CALL CLRXY()
       CALL SETXOR(0)
       END

************************************************************************
************************************************************************

       SUBROUTINE DRAW11(TYPE)
       INTEGER          TYPE,SWITCH
       REAL             X1,Y1,X2,Y2
       REAL             WX1,WY1,WX2,WY2
       INTEGER          NBLUE,GREEN,LBLUE,RED,VIOLET,BROWN,
     1                  WHITE,GRAY,BBLUE,BGREEN,BBBLUE,BRED,
     1                  BVIOLT,YELLOW,BWHITE
       COMMON/COLORS/   NBLUE,GREEN,LBLUE,RED,VIOLET,BROWN,
     1                  WHITE,GRAY,BBLUE,BGREEN,BBBLUE,BRED,
     1                  BVIOLT,YELLOW,BWHITE
       COMMON/ARRAS/    INTARR(100,10),REARR(100,100,2),KOS



       CALL WAIT4()
       CALL RESETW()

       INTARR(KOS,1) = TYPE

30     CALL READLO(X1,Y1,SWITCH)
       CALL DISXY(X1,Y1)
       CALL MOVHCA(X1,Y1)
       CALL MOVABS(X1,Y1)
       IF (SWITCH .EQ. 4 .OR. SWITCH .EQ. 132) THEN
          CALL CLRXY()
          CALL FIXLN()
          CALL WAIT4()
          INTARR(KOS,2) = 1
          REARR(KOS,1,1) = X1
          REARR(KOS,1,2) = Y1
          CALL DELHCU()
          CALL SETXOR(1)
50        CALL READLO(X2,Y2,SWITCH)
          CALL DISXY(X2,Y2)
          CALL RLNABS(X2,Y2)
          IF (SWITCH .EQ. 4 .OR. SWITCH .EQ. 132) THEN
             CALL CLRXY()
             CALL WAIT4()
```

```
            CALL DELLN()
            CALL SETXOR(0)
            CALL FIXLN()
            CALL LNABS(X2,Y2)
            INTARR(KOS,2) = INTARR(KOS,2) + 1
            REARR(KOS,INTARR(KOS,2),1) = X2
            REARR(KOS,INTARR(KOS,2),2) = Y2
            CALL SETXOR(1)
         ELSEIF( SWITCH .EQ. 2 .OR. SWITCH .EQ. 130) THEN
                CALL DELLN()
                CALL WAIT2()
                CALL SETXOR(0)
                IF (INTARR(KOS,2) .LT. 2) THEN
                   CONTINUE
                ELSE
                   IF (TYPE .LE. 9 .OR. (TYPE .EQ. 11) ) THEN

                       CALL DRAWLN(TYPE)
                   ELSEIF (TYPE .EQ. 12 .OR. TYPE .EQ. 13) THEN
                       CALL DRBODY(TYPE)
                   ELSEIF (TYPE .EQ. 14) THEN
                       CALL DRAWLN(TYPE)
                       CALL CONTUR()
                   ENDIF
                   KOS = KOS + 1
                ENDIF
                GOTO 10000
         ENDIF
         GOTO 50
      ELSEIF (SWITCH .EQ. 2 .OR. SWITCH .EQ. 130) THEN
         CALL WAIT2()
         GOTO 10000
      ENDIF
      GOTO 30
10000 CALL CLRXY()
      CALL SETXOR(0)
      END

************************************************************************
************************************************************************

      SUBROUTINE SUBBOX()
      INTEGER           NBLUE,GREEN,LBLUE,RED,VIOLET,BROWN,
     1                  WHITE,GRAY,BBLUE,BGREEN,BBBLUE,BRED,
     1                  BVIOLT,YELLOW,BWHITE
      COMMON/COLORS/    NBLUE,GREEN,LBLUE,RED,VIOLET,BROWN,
     1                  WHITE,GRAY,BBLUE,BGREEN,BBBLUE,BRED,
     1                  BVIOLT,YELLOW,BWHITE
      COMMON/ARRAS/      INTARR(100,10),REARR(100,100,2),KOS

      CALL SETHAT(1)
      CALL SETCOL(BBLUE)
      CALL SETCOL(BBLUE)
      XS = REARR(KOS,1,1)
      YS = REARR(KOS,1,2)
      XL = REARR(KOS,2,1)
      YL = REARR(KOS,2,2)
      CALL BAR(XS,YS,XL,YL)
10000 END


**********************************************************************
**********************************************************************


      SUBROUTINE DRBODY(TYPE)
      INTEGER          TYPE,SWITCH
      REAL             X1,Y1
      INTEGER          NBLUE,GREEN,LBLUE,RED,VIOLET,BROWN,
     1                 WHITE,GRAY,BBLUE,BGREEN,BBBLUE,BRED,
     1                 BVIOLT,YELLOW,BWHITE
      COMMON/ARRAS/    INTARR(100,10),REARR(100,100,2),KOS
      COMMON/COLORS/   NBLUE,GREEN,LBLUE,RED,VIOLET,BROWN,
     1                 WHITE,GRAY,BBLUE,BGREEN,BBBLUE,BRED,
     1                 BVIOLT,YELLOW,BWHITE


      INTARR(KOS,3) = 0
      CALL RESETW()
      CALL SETXOR(0)
70    CALL READLO(X1,Y1,SWITCH)
      CALL DISXY(X1,Y1)
      CALL MOVHCA(X1,Y1)
      IF (SWITCH .EQ. 4 .OR. SWITCH .EQ. 132) THEN
         CALL DELHCU()
         CALL WAIT4()
         INTARR(KOS,2) = INTARR(KOS,2) + 1
         REARR(KOS,INTARR(KOS,2),1) = X1
         REARR(KOS,INTARR(KOS,2),2) = Y1
         INTARR(KOS,3) = INTARR(KOS,3) + 1
         GOTO 70
      ELSEIF (SWITCH .EQ. 2 .OR. SWITCH .EQ. 130) THEN
         GOTO 900
      ENDIF
      GOTO 70
900   CALL CLRXY()
      CALL DRAWLN(TYPE)
      CALL SUBFIL(TYPE)
10000 END

**********************************************************************
**********************************************************************
```

```
        SUBROUTINE SUBFIL(TYPE)

        INTEGER           TYPE
        REAL              X,Y

        INTEGER           NBLUE,GREEN,LBLUE,RED,VIOLET,BROWN,
     1                    WHITE,GRAY,BBLUE,BGREEN,BBBLUE,BRED,
     1                    BVIOLT,YELLOW,BWHITE
        COMMON/COLORS/    NBLUE,GREEN,LBLUE,RED,VIOLET,BROWN,
     1                    WHITE,GRAY,BBLUE,BGREEN,BBBLUE,BRED,
     1                    BVIOLT,YELLOW,BWHITE
        COMMON/ARRAS/     INTARR(100,10),REARR(100,100,2),KOS
        COMMON/WORLDC/    XB,YB,XT,YT,ILEVEL

        ISTR = INTARR(KOS,2) - INTARR(KOS,3) + 1
        IEND = INTARR(KOS,2)
        DO 100 K = ISTR , IEND
           X = REARR(KOS,K,1)
           Y = REARR(KOS,K,2)
           IF (X .LT. XT .AND. X .GT. XB .AND.
     1         Y .LT. YT .AND. Y .GT. YB) THEN
               CALL MOVABS(X,Y)
               IF (TYPE .EQ. 12) THEN
                  CALL SETHAT(5)
                  CALL SETHAT(5)
                  CALL FLOOD2(NBLUE,NBLUE)
                  CALL FLOOD2(NBLUE,NBLUE)
               ELSEIF (TYPE .EQ. 13) THEN
                  CALL SETHAT(3)
                  CALL SETHAT(3)
                  CALL FLOOD2(BGREEN,BGREEN)
                  CALL FLOOD2(BGREEN,BGREEN)
               ELSEIF(TYPE .EQ. 15) THEN
                  CALL SETHAT(1)
                  CALL SETHAT(1)
                  CALL FLOOD(BRED)
                  CALL FLOOD(BRED)
               ENDIF
           ENDIF
100     CONTINUE
10000   END

************************************************************************
************************************************************************

        SUBROUTINE SUBCKF()
        REAL              X,Y
        INTEGER           TYPE,LIMIT
```

```
      COMMON/ARRAS/     INTARR(100,10),REARR(100,100,2),KOS

      DO 100 I = 6,10
         CALL MOVABS(REARR(KOS,I,1),REARR(KOS,I,2))
100   CONTINUE
10000 END

**************************************************************************
**************************************************************************
**************************************************************************

      SUBROUTINE CURVE()

      COMMON/WORLDC/   XB,YB,XT,YT,ILEVEL
      COMMON/DERDAT/   RCARS(40,13),NCARS,UWIND,UANGL,PATM,
     1                 TATM,UTRAIN,RCURV,ICARDR,ITRUCK,IAXLE,
     1                 CENTER,D1X,D1Y,D2X,D2Y,
     1                 USTR,UEND,DX,DY,IDTRK,ISPEED,CENX,CENY,
     1                 NUMTNK,DERAIL,DERTYP,ICAR,ATMOSP
      COMMON/BLK/      IBLACK
      COMMON/TRK/      IWIDTH


      CALL SETVIE(0.,0.,.0001,.0001,0,GREEN)
      CALL SETVIE(0.,0.,.746875,.85,-1,-1)
      CALL SETWOR(XB,YT,XT,YB)
      CALL  SETLEV()
      IF (IWIDTH .EQ. 1) THEN
         CALL SETCOL(IBLACK)
         CALL SETCOL(IBLACK)
         CALL SETLNS(3)
         CALL SETLNS(3)
         CALL SETLNW(3*ILEVEL)
         CALL SETLNW(3*ILEVEL)
      ELSEIF (IWIDTH .EQ. 2) THEN
         CALL SETCOL(IBLACK)
         CALL SETCOL(IBLACK)
         CALL SETLNS(3)
         CALL SETLNS(3)
         CALL SETLNW(5*ILEVEL)
         CALL SETLNW(5*ILEVEL)
      ENDIF
      CALL RESETW()
      RINC = USTR
      AINC = .02
      X = RCURV*COS(USTR)
      Y = RCURV*SIN(USTR)
      D1X = X+CENX
      D1Y = Y-CENY
      CALL MOVABS(X+CENX,CENY-Y)
      CALL LNABS(X+CENX,CENY-Y)
4300  IF (RINC .LE. UEND) THEN
```

```
            RINC = RINC + AINC
            X = RCURV*COS(RINC)
            Y = RCURV*SIN(RINC)
            CALL LNABS(X+CENX,CENY-Y)
            GOTO 4300
         ENDIF
         D2X = X+CENX
         D2Y = Y-CENY

10000    END

*************************************************************************
*************************************************************************

         SUBROUTINE TANGET()

         COMMON/WORLDC/   XB,YB,XT,YT,ILEVEL
         COMMON/DERDAT/   RCARS(40,13),NCARS,UWIND,UANGL,PATM,
     1                    TATM,UTRAIN,RCURV,ICARDR,ITRUCK,IAXLE,
     1                    CENTER,D1X,D1Y,D2X,D2Y,
     1                    USTR,UEND,DX,DY,IDTRK,ISPEED,CENX,CENY,
     1                    NUMTNK,DERAIL,DERTYP,ICAR,ATMOSP
         COMMON/BLK/      IBLACK
         COMMON/TRK/      IWIDTH

         CALL SETVIE(0.,0.,.0001,.0001,0,GREEN)
         CALL SETVIE(0.,0.,.746875,.85,-1,-1)
         CALL SETWOR(XB,YT,XT,YB)
         CALL SETLEV()
         IF (IWIDTH .EQ. 1) THEN
            CALL SETCOL(IBLACK)
            CALL SETCOL(IBLACK)
            CALL SETLNS(3)
            CALL SETLNS(3)
            CALL SETLNW(3*ILEVEL)
            CALL SETLNW(3*ILEVEL)
         ELSEIF (IWIDTH .EQ. 2) THEN
            CALL SETCOL(IBLACK)
            CALL SETCOL(IBLACK)
            CALL SETLNS(3)
            CALL SETLNS(3)
            CALL SETLNW(5*ILEVEL)
            CALL SETLNW(5*ILEVEL)
         ENDIF
         IF (IF ILEVEL .EQ. 1) THEN
              CALL MOVABS(D1X,D1Y)
              CALL LNABS(D2X,D2Y)
         ELSE
            CALL CHECK2(D1X,D1Y,D2X,D2Y)
         ENDIF

10000    END
```

```
************************************************************************
************************************************************************


      SUBROUTINE CHECK3(X1,Y1,X2,Y2)
      REAL             X1,Y1,X2,Y2
      LOGICAL          FLAG
      REAL           . X,Y,XX1,YY1,XX2,YY2,M,B

      COMMON/WORLDC/   XB,YB,XT,YT,ILEVEL


      XX1 = 0.
      YY1 = 0.
      XX2 = 0.
      YY2 = 0.

      IF ((X1 .EQ. X2) .OR. (Y1 .EQ. Y2)) GOTO 10000
      CALL SETLEV()
      IF ((X1 .LT. XB .OR. X1 .GT. XT .OR. Y1 .LT. YB .OR.
     1     Y1 .GT. YT) .AND. (X2 .LT. XB .OR. X2 .GT. XT .OR.
     1     Y2 .LT. YB .OR. Y2 .GT. YT)) THEN
        M = (Y2-Y1)/(X2-X1)
        B = Y1 - M*X1
        Y = M*XT + B
        FLAG = .FALSE.
        IF (Y .GE. YB .AND. Y .LE. YT) THEN
          CALL MOVABS(XT,Y)
          FLAG = .TRUE.
        ENDIF
        X = (YB - B)/M
        IF (X .GE. XB .AND. X .LE. XT) THEN
          IF (FLAG) THEN
            CALL LNABS(X,YB)
            CALL PTABS(X,YB)
          ELSE
            CALL MOVABS(X,YB)
            FLAG = .TRUE.
          ENDIF
        ENDIF
        Y = M*XB + B
        IF (Y .GE. YB .AND. Y .LE. YT) THEN
          IF (FLAG) THEN
            CALL LNABS(XB,Y)
            CALL PTABS(XB,Y)
          ELSE
            CALL MOVABS(XB,Y)
            FLAG = .TRUE.
          ENDIF
        ENDIF
```

```
           X = (YT - B)/M
           IF (X .GE. XB .AND. X .LE. XT) THEN
              CALL LNABS(X,YT)
              CALL PTABS(X,YT)
           ENDIF
        ENDIF
10000   END
```

```
*****************************************************************************
*****************************************************************************
```

```
      SUBROUTINE SUBCAR
$STORAGE:2
      REAL             TH,X1,Y1,DISTX,DISTY,HEIGX,HEIGY
      REAL             XFF(6),YFF(6),FX,FY,X(4),Y(4),AX(5),AY(5)
      INTEGER          NBLUE,GREEN,LBLUE,RED,VIOLET,BROWN,
     1                 WHITE,GRAY,BBLUE,BGREEN,BBBLUE,BRED,
     1                 BVIOLT,YELLOW,BWHITE
      COMMON/COLORS/   NBLUE,GREEN,LBLUE,RED,VIOLET,BROWN,
     1                 WHITE,GRAY,BBLUE,BGREEN,BBBLUE,BRED,
     1                 BVIOLT,YELLOW,BWHITE
      COMMON/ARRAS/    INTARR(100,10),REARR(100,100,2),KOS
      COMMON/DERDAT/   RCARS(40,13),NCARS,UWIND,UANGL,PATM,
     1                 TATM,UTRAIN,RCURV,ICARDR,ITRUCK,IAXLE,
     1                 CENTER,D1X,D1Y,D2X,D2Y,
     1                 USTR,UEND,DX,DY,IDTRK,ISPEED,CENX,CENY,
     1                 NUMTNK,DERAIL,DERTYP,ICAR,ATMOSP
      COMMON/WORLDC/   XB,YB,XT,YT,ILEVEL
      COMMON/DIRC/     DIRECT

C**** SET UP CAR DIMENSIONS
      X1 = RCARS(ICAR,6)
      Y1 = RCARS(ICAR,7)
      X(1) = -10.0
      Y(1) = -2.
      X(2) =  10.0
      Y(2) = -2.
      X(3) =  10.0
      Y(3) =  2.
      X(4) = -10.0
      Y(4) =  2.

C**** SET UP FILL POINT DIMENSIONS

      XFF(1) = -9.0
      YFF(1) = -1.0
      XFF(2) =  0.0
      YFF(2) = -1.0
      XFF(3) =  9.0
      YFF(3) = -1.0
      XFF(4) =  9.0
      YFF(4) =  1.0
      XFF(5) =  0.0
      YFF(5) =  1.0
      XFF(6) = -9.0
      YFF(6) =  1.0

C**** SET UP FOR ARROW
      AX(1) = 0.
      AX(2) = -2.
      AX(3) = 0.
      AX(4) = 2.0
      AX(5) = 0.
```

```
        IF (DIRECT .LT. 0.) THEN
           AY(1) = -2.
           AY(2) = 0.
           AY(3) = -4.
           AY(4) = 0.
           AY(5) = -3.
        ELSE
           AY(1) = 2.
           AY(2) = 0.
           AY(3) = 4.
           AY(4) = 0.
           AY(5) = 3.
        ENDIF

C****   SET UP FOR ROTATION

        TH = RCARS(ICAR,8) - UANGL
        A =   COS(TH)
        B =   SIN(TH)
        C = -(SIN(TH))
        D =   COS(TH)
        INTARR(KOS,2) = 0

C****   CALCULATE THE POINTS OF CAR
C****   AND STORE POINTS IN ARRAY

        DO 300 I = 1,4
            TX = A*X(I) + C*Y(I)
            TY = B*X(I) + D*Y(I)
            INTARR(KOS,2) = INTARR(KOS,2) + 1
            REARR(KOS,INTARR(KOS,2),1) = TX+X1
            REARR(KOS,INTARR(KOS,2),2) = Y1 - TY
300     CONTINUE
        INTARR(KOS,2) = INTARR(KOS,2) + 1
        REARR(KOS,INTARR(KOS,2),1) = REARR(KOS,1,1)
        REARR(KOS,INTARR(KOS,2),2) = REARR(KOS,1,2)

C****   NOW CALCULATE THE FLOOD POINTS
C****   AND STORE POINTS IN ARRAY

        DO 350 I = 1,6
            FX = (A*XFF(I) + C*YFF(I)) + X1
            FY = Y1 - (B*XFF(I) + D*YFF(I))
            INTARR(KOS,2) = INTARR(KOS,2) + 1
            REARR(KOS,INTARR(KOS,2),1) = FX
            REARR(KOS,INTARR(KOS,2),2) = FY
350     CONTINUE

C
C****   CALCULATE RUPTURE POINTS
```

```
C
      DISTX = 2.0
      DISTY = -5.0
      HIEGX = -2.0
      HIEGY = 5.0
      TX = A*DISTX + C*DISTY
      TY = B*DISTX + D*DISTY
      INTARR(KOS,2) = INTARR(KOS,2) + 1
      REARR(KOS,INTARR(KOS,2),1) = TX + X1
      REARR(KOS,INTARR(KOS,2),2) = Y1 - TY
      TX = A*HIEGX + C*HIEGY
      TY = B*HIEGX + D*HIEGY
      INTARR(KOS,2) = INTARR(KOS,2) + 1
      REARR(KOS,INTARR(KOS,2),1) = TX + X1
      REARR(KOS,INTARR(KOS,2),2) = Y1 - TY


C
C****  CALCULATE ARROW POINTS
C
      DO 500 I = 1,5
         TX = A*AX(I) + C*AY(I)
         TY = B*AX(I) + D*AY(I)
         INTARR(KOS,2) = INTARR(KOS,2) + 1
         REARR(KOS,INTARR(KOS,2),1) = TX+X1
         REARR(KOS,INTARR(KOS,2),2) = Y1 - TY
500   CONTINUE
      CALL SUBDRC()
10000 END


************************************************************************
************************************************************************

      SUBROUTINE SUBDRC()

      REAL              RX,RY,X1,Y1
      INTEGER           NBLUE,GREEN,LBLUE,RED,VIOLET,BROWN,
     1                  WHITE,GRAY,BBLUE,BGREEN,BBBLUE,BRED,
     1                  BVIOLT,YELLOW,BWHITE
      COMMON/COLORS/    NBLUE,GREEN,LBLUE,RED,VIOLET,BROWN,
     1                  WHITE,GRAY,BBLUE,BGREEN,BBBLUE,BRED,
     1                  BVIOLT,YELLOW,BWHITE
      COMMON/ARRAS/     INTARR(100,10),REARR(100,100,2),KOS
      COMMON/WORLDC/    XB,YB,XT,YT,ILEVEL
      COMMON/DERDAT/    RCARS(40,13),NCARS,UWIND,UANGL,PATM,
     1                  TATM,UTRAIN,RCURV,ICARDR,ITRUCK,IAXLE,
     1                  CENTER,DSTRX,DSTRY,DENDX,DENDY,
     1                  USTR,UEND,DX,DY,IDTRK,ISPEED,CENX,CENY,
     1                  NUMTNK,DERAIL,DERTYP,ICAR,ATMOSP
      COMMON/BLK/       IBLACK
      COMMON/RUPT/      RPT(8)

      CALL SETXOR(0)
      CALL SETLEV()
      IF (RCARS(ICAR,1) .EQ. 1. .OR. RCARS(ICAR,1) .EQ. 3.) THEN
         CALL SETCOL(GRAY)
         CALL SETCOL(GRAY)
      ELSEIF (RCARS(ICAR,1) .EQ. 2.) THEN
         CALL SETCOL(YELLOW)
         CALL SETCOL(YELLOW)
      ENDIF
      CALL SETLNS(1)
      CALL SETLNS(1)
      CALL SETLNW(1)
      CALL SETLNW(1)
      CALL MOVABS(REARR(KOS,1,1),REARR(KOS,1,2))
      CALL PTABS(REARR(KOS,1,1),REARR(KOS,1,2))
      DO 100 I = 2, 5
         CALL LNABS(REARR(KOS,I,1),REARR(KOS,I,2))
         CALL PTABS(REARR(KOS,I,1),REARR(KOS,I,2))
100   CONTINUE
      IF (RCARS(ICAR,1) .NE. 2.) GOTO 450

      DO 200 I = 6,11
         RX = REARR(KOS,I,1)
         RY = REARR(KOS,I,2)
         IF (RX .LE. XT .AND. RX .GE. XB .AND. RY .LE. YT
     1       .AND. RY .GE. YB) THEN
            CALL MOVABS(RX,RY)
            IF (RCARS(ICAR,2) .EQ. 3.) THEN
               CALL SETHAT(3)
               CALL SETHAT(3)
            ELSEIF (RCARS(ICAR,2) .EQ. 2.) THEN
               CALL SETHAT(5)
               CALL SETHAT(5)
            ELSEIF (RCARS(ICAR,2) .EQ. 1.) THEN
               CALL SETHAT(4)
               CALL SETHAT(4)
            ENDIF
            IF (ILEVEL .EQ. 3) THEN
               CALL FLOOD2(YELLOW,YELLOW)
            ENDIF
         ENDIF
200   CONTINUE


C************
C******  DRAW RUPTURE
C************
      IF (RCARS(ICAR,10) .EQ. 1. .OR. RCARS(ICAR,10) .EQ. 2.) THEN
         CALL SETHAT(1)
         CALL SETLNS(1)
```

```
          IF (RPT(ICAR) .EQ. 1.0) THEN
             CALL SETLNW(3)
          ELSE
             CALL SETLNW(5)
          ENDIF
          CALL SETCOL(IBLACK)
          CALL SETCOL(IBLACK)

          X1 = REARR(KOS,12,1)
          Y1 = REARR(KOS,12,2)
          X2 = REARR(KOS,13,1)
          Y2 = REARR(KOS,13,2)
          CALL MOVABS(X1,Y1)
          CALL LNABS(X2,Y2)
        ENDIF

C**************
C*****  DRAW ROLL MARK
C**************
450     IF (RCARS(ICAR,9) .GT. 0.) THEN
          CALL MOVABS(REARR(KOS,14,1),REARR(KOS,14,2))
          CALL PTABS(REARR(KOS,14,1),REARR(KOS,14,2))
          DO 500 I = 14,17
             CALL SETLNW(1)
             CALL SETLNS(1)
             CALL SETCOL(BLUE)
             CALL SETCOL(BLUE)
             X = REARR(KOS,I,1)
             Y = REARR(KOS,I,2)
             CALL LNABS(X,Y)
             CALL PTABS(X,Y)
500       CONTINUE
          CALL LNABS(REARR(KOS,14,1),REARR(KOS,14,2))
          CALL PTABS(REARR(KOS,14,1),REARR(KOS,14,2))
          X = REARR(KOS,18,1)
          Y = REARR(KOS,18,2)
          IF ((ILEVEL .EQ. 3) .AND. (X .GT. XB .AND. X .LT. XT
     1         .AND. Y .GT. YB .AND. Y .LT. YT)) THEN
               CALL MOVABS(X,Y)
               CALL SETHAT(0)
               CALL FLOOD2(IBLACK,BLUE)
          ENDIF
        ENDIF

        CALL RESETW()
        END

****************************************************************************
****************************************************************************
```

```
      SUBROUTINE  SUBRDR()
$STORAGE:2
      INTEGER           TYPE
      INTEGER           NUMCR
      INTEGER           NBLUE,GREEN,LBLUE,RED,VIOLET,BROWN,
     1                  WHITE,GRAY,BBLUE,BGREEN,BBBLUE,BRED,
     1                  BVIOLT,YELLOW,BWHITE
      COMMON/COLORS/    NBLUE,GREEN,LBLUE,RED,VIOLET,BROWN,
     1                  WHITE,GRAY,BBLUE,BGREEN,BBBLUE,BRED,
     1                  BVIOLT,YELLOW,BWHITE
      COMMON/ARRAS/     INTARR(100,10),REARR(100,100,2),KOS
      COMMON/DERDAT/    RCARS(40,13),NCARS,UWIND,UANGL,PATM,
     1                  TATM,UTRAIN,RCURV,ICARDR,ITRUCK,IAXLE,
     1                  CENTER,D1X,D1Y,D2X,D2Y,
     1                  USTR,UEND,DX,DY,IDTRK,ISPEED,CENX,CENY,
     1                  NUMTNK,DERAIL,DERTYP,ICAR,ATMOSP
      COMMON/RESULT1/  IBLAST(8),BLAST(8,7),IFLARE(8),FLAR(8,3),
     1                  IPUFF(8),PUF(8,3),ISP(8),SP(8,6),
     1                  IPLUM(8),PLM(8,2),PLMM(300,2),
     1                  IFIRE(8),FIRE(8,4),RUPDAT(8,5),IPLM(8),IPUF(8)
      COMMON/DET/       IDETAL(7)
      COMMON/MODE/      IMODE
      COMMON/TRKK/      ITRACK


************************************************
      CALL SETCOL(0)
      CALL SETCOL(0)
      CALL SETLNS(1)
      CALL SETLNS(1)
      CALL SETLNW(1)
      CALL SETLNW(1)
      CALL SETXOR(0)
      CALL SETVIE(0.,0.,.746875,.85,0,GREEN)
      CALL RESETW()
      CALL DELHCU()
      I = KOS
      DO 300 KOS = 3, I
         TYPE = INTARR(KOS,1)
         IF (TYPE .EQ. 12 .OR. TYPE .EQ. 13) THEN
            CALL DRAWLN(TYPE)
            CALL SUBFIL(TYPE)
         ENDIF
300   CONTINUE

      DO 400 KOS = 3, I
        TYPE = INTARR(KOS,1)
        IF (TYPE .GT. 4 .AND. TYPE .LT. 10) THEN
            CALL DRAWLN(TYPE)
        ELSEIF (TYPE .EQ. 11 .OR. TYPE .EQ. 14) THEN
            CALL DRAWLN(TYPE)
        ENDIF
```

```
400     CONTINUE

        DO 500 KOS = 3, I
          TYPE = INTARR(KOS,1)
          IF (TYPE .EQ. 10) THEN
              CALL SUBBOX()
          ENDIF
500     CONTINUE

        IF (ITRACK .EQ. 2.) THEN
            IF (CENTER .EQ. 1) THEN
              CALL TANGET()
            ELSE
              CALL CURVE()
            ENDIF
        ENDIF
        ICAR = 0
        DO 600 K = 1,NCARS
          ICAR = ICAR + 1
          IF (RCARS(I,1) .NE. 2.0) THEN
             CALL SUBCAR()
           ENDIF
600     CONTINUE
        ICAR = 0
        DO 601 K = 1,NCARS
          ICAR = ICAR + 1
          IF (RCARS(I,1) .EQ. 2.0) THEN
             CALL SUBCAR()
           ENDIF
601     CONTINUE
        IF (IMODE .EQ. 2) THEN
           ICAR = 0
           DO 700 K = 1,NCARS
              IF (RCARS(K,1) .EQ. 2) THEN
                     ICAR = ICAR + 1
                     IF (IDETAL(1) .EQ. 1 .OR. IDETAL(2) .EQ. 1) THEN
                        CALL DBLAST(K)
                     ENDIF
                     IF (IDETAL(3) .EQ. 1) THEN
                        CALL DFIRE(K)
                     ENDIF
                     IF (IDETAL(4) .EQ. 1) THEN
                        CALL DPUFF(K)
                     ENDIF
                     IF (IDETAL([illegible] .EQ. 1) THEN
                        CALL DPLM(K)
                     ENDIF
                     IF (IDETAL(6) .EQ. 1) THEN
                        CALL DFLAR(K)
                     ENDIF
                     IF (IDETAL(7) .EQ. 1) THEN
                        CALL DSPLL(K)
```

```
                ENDIF
             ENDIF
700     CONTINUE
      ENDIF
      KOS = I
10000 END

************************************************************************
************************************************************************


      SUBROUTINE  DRAWLN(TYPE)
      INTEGER            TYPE,NUMLNS
      REAL               X1,Y1,X2,Y2
      INTEGER            NBLUE,GREEN,LBLUE,RED,VIOLET,BROWN,
     1                   WHITE,GRAY,BBLUE,BGREEN,BBBLUE,BRED,
     1                   BVIOLT,YELLOW,BWHITE
      COMMON/ARRAS/      INTARR(100,10),REARR(100,100,2),KOS
      COMMON/COLORS/     NBLUE,GREEN,LBLUE,RED,VIOLET,BROWN,
     1                   WHITE,GRAY,BBLUE,BGREEN,BBBLUE,BRED,
     1                   BVIOLT,YELLOW,BWHITE
      COMMON/WORLDC/     XB,YB,XT,YT,ILEVEL
      COMMON/BLK/        IBLACK


      CALL RESETW()
      CALL SETLEV()
      IF (TYPE .EQ. 12 .OR. TYPE .EQ. 13) THEN
        NUMLNS = INTARR(KOS,2) - INTARR(KOS,3)
      ELSE
        NUMLNS = INTARR(KOS,2)
      ENDIF

      IF     (TYPE .EQ. 5) THEN
        CALL SETCOL(IBLACK)
        CALL SETCOL(IBLACK)
        CALL SETLNS(3)
        CALL SETLNS(3)
        CALL SETLNW(3*ILEVEL)
        CALL SETLNW(3*ILEVEL)
      ELSEIF (TYPE .EQ. 6) THEN
        CALL SETCOL(IBLACK)
        CALL SETCOL(IBLACK)
        CALL SETLNS(3)
        CALL SETLNS(3)
        CALL SETLNW(5*ILEVEL)
        CALL SETLNW(5*ILEVEL)
      ELSEIF (TYPE .EQ. 7) THEN
        CALL SETCOL(BROWN)
        CALL SETCOL(BROWN)
        CALL SETLNS(1)
        CALL SETLNS(1)
```

```
          IF (ILEVEL .EQ. 1) THEN
             CALL SETLNW(5)
             CALL SETLNW(5)
          ELSEIF (ILEVEL .EQ. 2) THEN
             CALL SETLNW(9)
             CALL SETLNW(9)
          ELSE
             CALL SETLNW(13)
             CALL SETLNW(13)
          ENDIF
       ELSEIF (TYPE .EQ. 8) THEN
          CALL SETCOL(BROWN)
          CALL SETCOL(BROWN)
          CALL SETCOL(BROWN)
          CALL SETLNS(1)
          CALL SETLNS(1)
          IF (ILEVEL .EQ. 1) THEN
             CALL SETLNW(3)
             CALL SETLNW(3)
          ELSEIF (ILEVEL .EQ. 2) THEN
             CALL SETLNW(5)
             CALL SETLNW(5)
          ELSE
             CALL SETLNW(9)
             CALL SETLNW(9)
          ENDIF
       ELSEIF (TYPE .EQ. 9) THEN
          CALL SETCOL(BROWN)
          CALL SETCOL(BROWN)
          CALL SETLNS(1)
          CALL SETLNS(1)
          IF (ILEVEL .EQ. 1) THEN
             CALL SETLNW(1)
             CALL SETLNW(1)
          ELSEIF (ILEVEL .EQ. 2) THEN
             CALL SETLNW(3)
             CALL SETLNW(3)
          ELSE
             CALL SETLNW(7)
             CALL SETLNW(7)
          ENDIF
       ELSEIF (TYPE .EQ. 11) THEN
          CALL SETCOL(BBBLUE)
          CALL SETCOL(BBBLUE)
          CALL SETLNS(1)
          CALL SETLNS(1)
          CALL SETLNW(1)
          CALL SETLNW(1)
       ELSEIF (TYPE .EQ. 12) THEN
          CALL SETCOL(NBLUE)
          CALL SETCOL(NBLUE)
          CALL SETLNS(1)
```

```
            CALL SETLNS(1)
            CALL SETLNW(1)
            CALL SETLNW(1)
         ELSEIF (TYPE .EQ. 13) THEN
            CALL SETCOL(BGREEN)
            CALL SETCOL(BGREEN)
            CALL SETLNS(1)
            CALL SETLNS(1)
            CALL SETLNW(1)
            CALL SETLNW(1)
         ELSEIF (TYPE .EQ. 14) THEN
            CALL SETCOL(VIOLET)
            CALL SETCOL(VIOLET)
            CALL SETLNS(1)
            CALL SETLNS(1)
            CALL SETLNW(1)
            CALL SETLNW(1)
         ENDIF
         IF (ILEVEL .EQ. 1) THEN
            DO 70 I = 2,NUMLNS
               X1 = REARR(KOS,I-1,1)
               Y1 = REARR(KOS,I-1,2)
               X2 = REARR(KOS,I,1)
               Y2 = REARR(KOS,I,2)
               CALL MOVABS(X1,Y1)
               CALL PTABS(X1,Y1)
               CALL LNABS(X2,Y2)
70          CONTINUE
         ELSE
            X1 = REARR(KOS,1,1)
            Y1 = REARR(KOS,1,2)
            CALL MOVABS(X1,Y1)
            CALL PTABS(X1,Y1)
            DO 60 I = 2, NUMLNS
               X1 = REARR(KOS,I-1,1)
               Y1 = REARR(KOS,I-1,2)
               X2 = REARR(KOS,I,1)
               Y2 = REARR(KOS,I,2)
               CALL CHECK2(X1,Y1,X2,Y2)
60          CONTINUE
         ENDIF
10000    END

************************************************************************
************************************************************************

         SUBROUTINE CHECK2(X1,Y1,X2,Y2)
         REAL           X1,Y1,X2,Y2
         LOGICAL        FLAG
         REAL           X,Y,M,B

         COMMON/WORLDC/   XB,YB,XT,YT,ILEVEL
```

```
      IF ((X1 .EQ. X2) .OR. (Y1 .EQ. Y2)) THEN
         CALL PTABS(X1,Y1)
         CALL LNABS(X2,Y2)
         GOTO 10000
      ENDIF
      IF ( (X1 .GE. XB .AND. X1 .LE. XT .AND. Y1 .GE. YB
     1        .AND. Y1 .LE. YT) .OR. (X2 .GE. XB .AND.
     1         X2 .LE. XT .AND. Y2 .GE. YB .AND. Y2 .LE. YT)) THEN
         CALL PTABS(X1,Y1)
         CALL LNABS(X2,Y2)
         GOTO 10000
      ENDIF
      CALL SETLEV()
      M = (Y2-Y1)/(X2-X1)
      B = Y1 - M*X1
      Y = M*XT + B
      FLAG = .FALSE.
      IF (Y .GE. MIN(Y1,Y2) .AND. Y .LE. MAX(Y1,Y2)) THEN
      IF (Y .GE. YB .AND. Y .LE. YT) THEN
         CALL MOVABS(XT,Y)
         FLAG = .TRUE.
      ENDIF
      ENDIF


      X = (YB - B)/M
      IF (X .GE. MIN(X1,X2) .AND. X .LE. MAX(X1,X2)) THEN
      IF (X .GE. XB .AND. X .LE. XT) THEN
         IF (FLAG) THEN
            CALL LNABS(X,YB)
            CALL PTABS(X,YB)
         ELSE
            CALL MOVABS(X,YB)
            FLAG = .TRUE.
         ENDIF
      ENDIF
      ENDIF


      Y = M*XB + B
      IF (Y .GE. MIN(Y1,Y2) .AND. Y .LE. MAX(Y2,Y1)) THEN
      IF (Y .GE. YB .AND. Y .LE. YT) THEN
         IF (FLAG) THEN
            CALL LNABS(XB,Y)
            CALL PTABS(XB,Y)
         ELSE
            CALL MOVABS(XB,Y)
            FLAG = .TRUE.
         ENDIF
      ENDIF
      ENDIF


      X = (YT - B)/M
      IF (X .GE. MIN(X1,X2) .AND. X .LE.  MAX(X1,X2)) THEN
      IF (X .GE. XB .AND. X .LE. XT) THEN
         CALL LNABS(X,YT)
         CALL PTABS(X,YT)
      ENDIF
      ENDIF

10000 END
```

```
**************************************************************************
**************************************************************************

      SUBROUTINE RCRS(INTT)
$STORAGE:2
      INTEGER          INTT,ITMP,IC

      COMMON/DERDAT/   RCARS(40,13),NCARS,UWIND,UANGL,PATM,
     1                 TATM,UTRAIN,RCURV,ICARDR,ITRUCK,IAXLE,
     1                 CENTER,D1X,D1Y,D2X,D2Y,
     1                 USTR,UEND,DX,DY,IDTRK,ISPEED,CENX,CENY,
     1                 NUMTNK,DERAIL,DERTYP,ICAR,ATMOSP
      COMMON/ARRAS/    INTARR(100,10),REARR(100,100,2),KOS
      COMMON/WINDS/    WSPEED
      COMMON/TRKK/     ITRACK
      COMMON/DIRC/     DIRECT
      COMMON/ROTATE/   BETA
      COMMON/RUPT/     RPT(8)
      COMMON/TRANS/    THETA
      COMMON/TRK/      IWIDTH
      COMMON/TANK/     ITCAR(40)


      ITMP = 0
      IC = 0


      OPEN(4,FILE='PRE.OUT',STATUS='OLD',FORM='UNFORMATTED')
      IF (INTT .NE. 0) THEN
      ENDIF
      READ(4)KOS
      DO 50 I = 1,KOS
         READ(4)(INTARR(I,K),K=1,10),
     1          ((REARR(I,J,K),K=1,2),J=1,100)
50    CONTINUE
      OPEN(5,FILE='PRE1.OUT',STATUS='OLD')
      READ(5,*) TATM,UANGL,RCURV,ICARDR,ITRUCK,IAXLE
      READ(5,*) CENTER,D1X,D1Y,D2X,D2Y
      READ(5,*) USTR,UEND,DX,DY,IDTRK,ISPEED,CENX,CENY
      READ(5,*) NUMTNK,DERAIL,DERTYP,ICAR,ATMOSP,ITRACK
      READ(5,*) BETA,DIRECT,THETA,IWIDTH
      IF (INTT .NE. 0) THEN
      ENDIF
      OPEN(2,FILE='CARDATA.DAT',STATUS='OLD')
      READ(2,*)SPEED,RCURV,ICARDR,ITRUCK
      READ(2,*)NCARS,WSPEED,UWIND,PATM,ATAM
      ICAR = 0
      DO 100 I = 1,NCARS
         READ(2,*)I,RCARS(I,1)
         IF (RCARS(I,1) .EQ. 2) THEN
            ICAR = ICAR + 1
            ITCAR(I) = ICAR
```

```
               READ(2,*)(RCARS(I,K),K=2,5)
            ENDIF
100      CONTINUE
         IF (INTT .NE. 0) THEN
         ENDIF
         OPEN(3,FILE='DERAIL.DAT',STATUS='OLD')
         IC = 0
         DO 200 I = 1,NCARS
            READ(3,*)ICAR,(RCARS(I,K),K=6,9)
            CALL CALCAR(I)
            IF (RCARS(I,1) .EQ. 2) THEN
               IC = IC + 1
               READ(3,*)ITMP,(RCARS(I,K),K=11,13)
               RCARS(I,10) = REAL(ITMP)
               IF (RCARS(I,10) .GT. 0.) RPT(IC) = 1.0
            ENDIF
200      CONTINUE

         IF (INTT .NE. 0) THEN
         ENDIF
         CLOSE(2)
         CLOSE(3)
         CLOSE(4)
10000    END
```

```
      SUBROUTINE SUBDER()
$STORAGE:2
      CHARACTER        JUNK
      COMMON/DERDAT/   RCARS(40,13),NCARS,UWIND,UANGL,PATM,
     1                 TATM,UTRAIN,RCURV,ICARDR,ITRUCK,IAXLE,
     1                 CENTER,D1X,D1Y,D2X,D2Y,
     1                 USTR,UEND,DX,DY,IDTRK,ISPEED,CENX,CENY,
     1                 NUMTNK,DERAIL,DERTYP,ICAR,ATMOSP
      COMMON/TRKK/     ITRACK
      COMMON/RUPT/      RPT(8)

      IF (ITRACK .NE. 2.) THEN
         CALL FIXLN()
         CALL SETVIE(.76,.5,.99,.75,0,1)
         CALL SETWOR(0.,10.,10.,0.)
         CALL MOVTCA(1.,2.)
         CALL TEXT('/DERAILMENT/')
         CALL MOVTCA(1.,4.)
         CALL TEXT('/TRACK MUST BE/')
         CALL MOVTCA(1.,6.)
         CALL TEXT('/ENTERED!/')
         CALL MOVTCA(1.,10.)
         CALL TEXT('/<C> TO CONTINUE/')
         CALL DELTCU()
         CALL PROMPT()
         READ(*,'(A)',END=3,ERR=3)JUNK
3        CONTINUE
         CALL CPROM()
         CALL DRICON()
         GOTO 10000
      ENDIF


      IF (DERAIL .EQ. 2.) THEN
         CALL FIXLN()
         CALL SETVIE(.76,.4,.99,.75,0,1)
         CALL SETWOR(0.,14.,10.,0.)
         CALL MOVTCA(1.,2.)
         CALL TEXT('/DERAILMENT/')
         CALL MOVTCA(1.,4.)
         CALL TEXT('/MECHANICS HAS/')
         CALL MOVTCA(1.,6.)
         CALL TEXT('/BEEN ENTERED!/')
         CALL MOVTCA(1.,10.)
         CALL TEXT('/<C> TO CONTINUE/')
         CALL MOVTCA(1.,12.)
         CALL TEXT('/<Q> TO QUIT/')
         CALL DELTCU()
         CALL PROMPT()
         READ(*,'(A)',END=4,ERR=4)JUNK
4        CONTINUE
         CALL CPROM()
```

```
           IF (JUNK .NE. 'C' .AND. JUNK .NE. 'c') THEN
              CALL DRICON()
              GOTO 10000
           ENDIF
        ENDIF


        CALL CLOSEG()
        WRITE(*,*)' 2J'
        IF (NCARS .GE. 1) DERAIL = 2.
        DO 1 KK = 1,8
           RPT(KK) = 0.
1       CONTINUE
        DO 100 I = 1, NCARS
           WRITE(*,'(A\)')' 2J4;25HDERAILMENT  MECHANICS'
           WRITE(*,'(A\)')'8;3HCAR # '
           WRITE(*,'(I3)')I
           IF (RCARS(I,2) .EQ. 1) THEN
              WRITE(*,'(A\)')' 8;14HCONTAINS PROPANE'
           ELSEIF (RCARS(I,2) .EQ. 2) THEN
              WRITE(*,'(A\)')' 8;14HCONTAINS CHLORINE'
           ELSEIF (RCARS(I,2) .EQ. 3) THEN
              WRITE(*,'(A\)')' 8;14HCONTAINS HEXANE'
           ELSEIF (RCARS(I,1) .EQ. 1) THEN
              WRITE(*,'(A\)')' 8;14HIS A LOCOMOTIVE'
           ELSEIF (RCARS(I,1) .EQ. 3) THEN
              WRITE(*,'(A\)')' 8;14HIS A BOX CAR'
           ENDIF
           WRITE(*,'(A\)')' 10;20HENTER X,Y POSITION'
20         WRITE(*,'(A,A\)')' 10;41H                         '
     1     ,' 10;41H-> 0,0 1m 10;41H-> 0m'
           READ(*,'(2F12.8)',END=20,ERR=20)RCARS(I,6),RCARS(I,7)
           IF (ABS(RCARS(I,6)).LT..01)RCARS(I,6) = RCARS(I,6) * 10**8
           IF (ABS(RCARS(I,7)).LT..01)RCARS(I,7) = RCARS(I,7) * 10**8
           IF (RCARS(I,6) .GT. 10000. .OR. RCARS(I,6) .LT. -10000. .OR.
     1         RCARS(I,7) .GT. 10000. .OR. RCARS(I,7) .LT.
     1         -10000.) GOTO 20
           CALL CALCAR(I)

           WRITE(*,'(A\)')' 12;20HENTER ANGLE OF CAR'
30         WRITE(*,'(A,A\)')' 12;40H                        '
     1     ,' 12;40H-> 01m12;40H-> 0m'
           READ(*,'(F12.8)',ERR=30,END=30)RCARS(I,8)
           IF (RCARS(I,8) .LT. .01) RCARS(I,8) = RCARS(I,8) * 10**8
           IF (RCARS(I,8).GT.360..OR. RCARS(I,8) .LT. 0.) GOTO 30
           RCARS(I,8) = RCARS(I,8) / 57.3

           WRITE(*,'(A\)')' 14;20HENTER ROLL ANGLE'
40         WRITE(*,'(A,A\)')' 14;38H                         '
     1     ,' 14;38H-> 01m14;38H-> 0m'
           READ(*,'(F12.8)',END=40,ERR=40)RCARS(I,9)
           IF (RCARS(I,9) .LT. .01) RCARS(I,9) = RCARS(I,9) * 10**8
```

```
            IF (RCARS(I,9).GT.360..OR. RCARS(I,9) .LT. 0.) GOTO 40
            RCARS(I,9) = RCARS(I,9) / 57.3


            IF (RCARS(I,1) .EQ. 2.) THEN
               WRITE(*,'(A\)')' 16;20HIS TANK RUPTURED ? 1.YES  0.NO'
50          WRITE(*,'(A,A\)')' 18;38H                       '
     1      ,' 18;38H-> 01m18;38H-> 0m'
               READ(*,'(I10)',END=50,ERR=50)IJUNK
               IF (IJUNK .GT. 1 .OR. IJUNK .LT. 0) GOTO 50
               IF (IJUNK .EQ. 1 ) THEN
                  RPT(I) = 1.0
            WRITE(*,'(A\)')' 19;20HENTER TYPE OF RUPTURE'
60          WRITE(*,'(A\)')' 20;26H                                  '
                  WRITE(*,'(A\)')' 1m20;26H->  0m'
                  READ(*,*,ERR=60,END=60)RCARS(I,10)
                  IF (RCARS(I,10) .GT. 2. .OR. RCARS(I,10) .LT. 1.)
     1                 GOTO 60
                  IF (RCARS(I,10) .EQ. 2.) THEN
            WRITE(*,'(A\)')' 22;20HENTER Diameter,Height,Distance'
70          WRITE(*,'(A\)')' 21;26H                                  '
                  WRITE(*,'(A\)')' 1m21;26H->  0m'
                  READ(*,*,ERR=70,END=70)(RCARS(I,KK),KK=11,13)
             IF (RCARS(I,11) .GT. .3  .OR. RCARS(I,11) .LE. 0. .OR.
     1       ABS(RCARS(I,12)) .GT. 1.5 .OR. ABS(RCARS(I,13)) .GT. 10.)
     1                   GOTO 70
                  ENDIF
             ELSE
                  RCARS(I,10) = 0.
             ENDIF
           ENDIF
100     CONTINUE

        WRITE(*,*)' 22;0H '
10000   END

**************************************************************************
```

```
      SUBROUTINE TRAK()
$STORAGE:2
      INTEGER          TYPE
      CHARACTER        JUNK
      COMMON/DERDAT/   RCARS(40,13),NCARS,UWIND,UANGL,PATM,
     1                 TATM,UTRAIN,RCURV,ICARDR,ITRUCK,IAXLE,
     1                 CENTER,D1X,D1Y,D2X,D2Y,
     1                 USTR,UEND,DX,DY,IDTRK,ISPEED,CENX,CENY,
     1                 NUMTNK,DERAIL,DERTYP,ICAR,ATMOSP
      COMMON/TRKK/     ITRACK
      COMMON/TRK/      IWIDTH

      CALL WAIT4()
      CALL FIXLN()
      CALL DWIN()
      CALL SETTEX(1,1,0,0)
      CALL SETTCL(12,2)
      CALL MOVTCA(1.,2.)
      CALL TEXT('/TRACK TYPE/')
      CALL MOVTCA(1.,2.)
      CALL TEXT('/__________/')
      CALL MOVTCA(1.,4.)
      CALL TEXT('/1. SINGLE/')
      CALL MOVTCA(1.,6.)
      CALL TEXT('/2. DOUBLE/')
      CALL DELTCU()
5     CALL PROMPT()
      READ(*,'(I10)',END=5,ERR=5)IDTRK
      IF (IDTRK .LT. 1 .OR. IDTRK .GT. 2) GOTO 5
      CALL CPROM()
      CALL FIXLN()
      CALL DWIN()
      CALL MOVTCA(1.,2.)
      CALL TEXT('/TRACK TYPE/')
      CALL MOVTCA(1.,2.)
      CALL TEXT('/__________/')
      CALL MOVTCA(1.,4.)
      CALL TEXT('/1. DERAILMENT/')
      CALL MOVTCA(1.,6.)
      CALL TEXT('/2. JOINED/')
      CALL MOVTCA(1.,8.)
      CALL TEXT('/3. OTHER/')
      CALL DELTCU()
20    CALL PROMPT()
      READ(*,'(I10)',END=20,ERR=20)TYPE
      IF (TYPE .LT. 1 .OR. TYPE .GT. 3) GOTO 20
      CALL CPROM()

      IF (ITRACK .EQ. 2. .AND. TYPE .EQ. 1) THEN

         CALL FIXLN()
         CALL DWIN()
         CALL MOVTCA(1.,2.)
         CALL TEXT('/DO YOU WANT/')
         CALL MOVTCA(1.,4.)
         CALL TEXT('/TO OVER-WRITE/')
         CALL MOVTCA(1.,6.)
         CALL TEXT('/TRACK?/')
         CALL MOVTCA(1.,10.)
         CALL TEXT('/<C> TO CONTINUE/')
         CALL DELTCU()
         CALL PROMPT()
         READ(*,'(A)',END=1,ERR=1)JUNK
1        CALL CPROM()
         IF (JUNK .NE. 'C' .AND. JUNK .NE. 'c') THEN
            CALL DRICON()
            GOTO 10000
         ELSE
            GOTO 2
         ENDIF
      ENDIF


      IF (TYPE .EQ. 2 .AND. ITRACK .EQ. 0) THEN
         CALL FIXLN()
         CALL DWIN()
         CALL MOVTCA(1.,2.)
         CALL TEXT('/DERAILMENT/')
         CALL MOVTCA(1.,4.)
         CALL TEXT('/TRACK MUST/')
         CALL MOVTCA(1.,6.)
         CALL TEXT('/BE INPUT!/')
         CALL MOVTCA(1.,10.)
         CALL TEXT('/<C> TO CONTINUE/')
         CALL DELTCU()
         CALL PROMPT()
         READ(*,'(A)',END=3,ERR=3)JUNK
3        CALL CPROM()
         CALL DRICON()
         GOTO 10000
      ENDIF


2     IF (TYPE .EQ. 1) THEN
         ITRACK = 2.
         UANGL = 0.
         IWIDTH = IDTRK
         CALL SUBDTK()
      ELSEIF (TYPE .EQ. 2) THEN
         CALL JOINED()
      ELSE
         CALL DRICON()
         CALL RESETW()
         CALL DRAW11(IDTRK+4)
      ENDIF
      CALL DRICON()

10000 END

************************************************************************
************************************************************************

      SUBROUTINE SUBDTK

      INTEGER          TYPE,SWITCH
      REAL             X1,Y1,X,Y
      REAL             WX1,WY1,WX2,WY2,RINC,AINC
      INTEGER          NBLUE,GREEN,LBLUE,RED,VIOLET,BROWN,
     1                 WHITE,GRAY,BBLUE,BGREEN,BBBLUE,BRED,
     1                 BVIOLT,YELLOW,BWHITE
      COMMON/COLORS/   NBLUE,GREEN,LBLUE,RED,VIOLET,BROWN,
     1                 WHITE,GRAY,BBLUE,BGREEN,BBBLUE,BRED,
     1                 BVIOLT,YELLOW,BWHITE
      COMMON/ARRAS/    INTARR(100,10),REARR(100,100,2),KOS
      COMMON/DERDAT/   RCARS(40,13),NCARS,UWIND,UANGL,PATM,
     1                 TATM,UTRAIN,RCURV,ICARDR,ITRUCK,IAXLE,
     1                 CENTER,D1X,D1Y,D2X,D2Y,
     1                 USTR,UEND,DX,DY,IDTRK,ISPEED,CENX,CENY,
     1                 NUMTNK,DERAIL,DERTYP,ICAR,ATMOSP
      COMMON/TRKK/     ITRACK
      COMMON/DIRC/     DIRECT
      COMMON/ROTATE/   BETA

      CALL FIXLN()
      CALL DWIN()
      CALL MOVTCA(1.,2.)
      CALL TEXT('/DERAILMENT TRACK/')
      CALL MOVTCA(1.,2.)
      CALL TEXT('/__________________/')
      CALL MOVTCA(1.,4.)
      CALL TEXT('/1. TANGENT/')
      CALL MOVTCA(1.,6.)
      CALL TEXT('/2. CURVED/')
      CALL DELTCU()
10    CALL PROMPT()
      READ(*,'(I10)',END=10,ERR=10)TYPE
      IF (TYPE .LT. 1 .OR. TYPE .GT. 2) GOTO 10
      CALL CPROM()
      IF (TYPE .EQ. 1) THEN
         CENTER = 1
         GOTO 2000
      ELSE
         CENTER = 2
         GOTO 3000
      ENDIF
      GOTO 10000
```

```
C********************
C********************
C**
C**  SECTION 2000 .............
C**
C**  CONSTRUCTION OF TANGENTIAL DERAILMENT TRACK STARTS HERE
C**
C********************
C********************
2000    CONTINUE
        CALL FIXLN()
        CALL DWIN()
        CALL MOVTCA(1.,2.)
        CALL TEXT('/CONSTRUCTING/')
        CALL MOVTCA(1.,4.)
        CALL TEXT('/DERAILMENT/')
        CALL MOVTCA(1.,6.)
        CALL TEXT('/TRACK;/')
        CALL MOVTCA(1.,10.)
        CALL TEXT('/LOCATE END#1/')
        CALL DELTCU()
        CALL RESETW()
2020    CALL READLO(X,Y,SWITCH)
        CALL DISXY(X,Y)
        CALL MOVHCA(X,Y)
        CALL ORGLOC(X,Y)
        IF (SWITCH .EQ. 4 .OR. SWITCH .EQ. 132) THEN
           CALL CLRXY()
           D1X = X
           D1Y = Y
           CALL FIXLN()
           CALL DWIN()
           CALL MOVTCA(1.,10.)
           CALL TEXT('/LOCATE END #2/')
           CALL DELHCU()
           CALL SETXOR(1)
           CALL RESETW()
           CALL FIXLN()
           CALL MOVABS(D1X,D1Y)
2030       CALL READLO(X,Y,SWITCH)
           CALL DISXY(X,Y)
           CALL ORGLOC(X,Y)
           CALL RLNABS(X,Y)
           IF (SWITCH .EQ. 4 .OR. SWITCH .EQ. 132) THEN
              CALL CLRXY()
              D2X = X
              D2Y = Y
              DDX = D2X - D1X
              DDY = D2Y - D1Y
              IF (DDX .EQ. 0.) DDX = .0001
              BETA = ACOS(ABS(DDX)/SQRT(DDX**2+DDY**2))
```

```
               IF (D1X .LT. D2X .AND. D1Y .GT. D2Y) THEN
                  UANGL = -(BETA)
               ELSEIF (D1X .LT. D2X .AND. D1Y .LT. D2Y) THEN
                  BETA = -(BETA)
                  UANGL = -(BETA)
               ELSEIF (D1X .GT. D2X .AND. D1Y .GT. D2Y) THEN
                  UANGL = -(BE¯A)
                  UANGL = BETA + 3.14159
               ELSEIF (D1X .GT. D2X .AND. D1Y .LT. D2Y) THEN
                  BETA = -(BETA)
                  UANGL = (BETA) + 3.14159
               ENDIF
               IF (D1Y .EQ. D2Y .AND. D1X .LT. D2X) THEN
                  BETA = 0.0
                  UANGL = BETA
               ELSEIF (D1Y .EQ. D2Y .AND. D1X .GT. D2X) THEN
                  BETA = 3.14159
                  UANGL = 0.0
               ENDIF
               IF (D1X .EQ. D2X .AND. D1Y .LT. D2Y) THEN
                  BETA = -1.57
                  UANGL = BETA
               ELSEIF (D1X .EQ. D2X .AND. D1Y .GT. D2Y) THEN
                  BETA = 1.57
                  UANGL = BETA
               ENDIF
               DIRECT = 1.0
               IF (D2X .LT. D1X) THEN
                  DIRECT = -1.0
                  X = D1X
                  D1X = D2X
                  D2X = X
                  Y = D1Y
                  D1Y = D2Y
                  D2Y = Y
               ENDIF
               IF (D1X .EQ. D2X) THEN
                  UANGL = UANGL + 3.14159
               ENDIF
               RCURV = 9999999.0
               CALL DELLN()
               CALL DRICON()
               CALL SUBRDR()
               CALL DTANGT()
               GOTO 10000
            ENDIF
            GOTO 2030
        ENDIF
        GOTO 2020

C********************
C********************
```

```
C**
C**  SECTION 3000 .............
C**
C**  CONTSTRUCTION OF CURVED DERAILMENT TRACK STARTS HERE
C**
C*******************
C*******************

3000    CONTINUE
        CALL FIXLN()
        CALL DWIN()
        CALL MOVTCA(1.,2.)
        CALL TEXT('/CONSTRUCTING/')
        CALL MOVTCA(1.,4.)
        CALL TEXT('/DERAILMENT/')
        CALL MOVTCA(1.,6.)
        CALL TEXT('/TRACK;/')
        CALL MOVTCA(1.,8.)
        CALL TEXT('/LOCATE CENTER/')
        CALL DELTCU()

        CALL RESETW()
3020    CALL READLO(X,Y,SWITCH)
        CALL DISXY(X,Y)
        CALL MOVHCA(X,Y)
        CALL ORGLOC(X,Y)
        IF (SWITCH .EQ. 4 .OR. SWITCH .EQ. 132) THEN
           CALL CLRXY()
           CENX = X
           CENY = Y
           CALL FIXLN()
           CALL DWIN()
           CALL MOVTCA(1.,2.)
           CALL MOVTCA(1.,2.)
           CALL TEXT('/ENTER RADIUS/')
           CALL MOVTCA(1.,4.)
           CALL TEXT('/OF CURVE/')
           CALL MOVTCA(1.,10.)
           CALL TEXT('/...Meters.../')
           CALL DELTCU()
3030       CALL PROMPT()
           READ(*,'(F12.8)',ERR=3030,END=3030)RCURV
           IF (RCURV .LT. .01) RCURV  = RCURV * 10**8
           IF (RCURV .LT. 0. .OR. RCURV .GT. 14000.) GOTO 3030
           CALL CPROM()
           CALL FIXLN()
           CALL DWIN()
           CALL MOVTCA(1.,2.)
           CALL TEXT('/ENTRANCE/')
           CALL MOVTCA(1.,4.)
           CALL TEXT('/ANGLE;/')
           CALL MOVTCA(1.,10.)
        CALL TEXT('/...Degrees CCW.../')
        CALL DELTCU()
3050    CALL PROMPT()
        READ(*,'(F12.8)',ERR=3050,END=3050)USTR
        IF (USTR  .LT. .01) USTR   = USTR  * 10**8
        IF (USTR .GT. 360. .OR. USTR .LT. 0.) GOTO 3050
        USTR = USTR / 57.3
        CALL CPROM()
        CALL FIXLN()
        CALL DWIN()
        CALL MOVTCA(1.,2.)
        CALL TEXT('/EXIT/')
        CALL MOVTCA(1.,4.)
        CALL TEXT('/ANGLE;/')
        CALL MOVTCA(1.,10.)
        CALL TEXT('/...Degrees CCW.../')
        CALL DELTCU()
3060    CALL PROMPT()
        READ(*,'(F12.8)',ERR=3060,END=3060)UEND
        IF (UEND  .LT. .01) UEND   = UEND  * 10**8
        IF (UEND .GT. 360. .OR. UEND .LT. 0.) GOTO 3060
        UEND = UEND / 57.3
        DIRECT =  -1.0
        IF (UEND .LT. USTR) THEN
           DIRECT = 1.0
           TMP = UEND
           UEND = USTR
           USTR = TMP
        ENDIF
        IF (DIRECT .LT. 0.) THEN
           D1X = CENX + RCURV*COS(USTR)
           D1Y = CENY - RCURV*SIN(USTR)
           D2X = CENX + RCURV*COS(UEND)
           D2Y = CENY - RCURV*SIN(UEND)
        ELSE
           D1X = CENX + RCURV*COS(UEND)
           D1Y = CENY - RCURV*SIN(UEND)
           D2X = CENX + RCURV*COS(USTR)
           D2Y = CENY - RCURV*SIN(USTR)
        ENDIF
        CALL CPROM()
        CALL DRICON()
        CALL SUBRDR()
        CALL DERTRK()
        GOTO 10000
     ENDIF
     GOTO 3020

10000  END

**************************************************************************
**************************************************************************
```

```
      SUBROUTINE DTANGT()

      INTEGER           SWITCH
      REAL              SLOPE,X1,Y1,X,Y,NX,M
      COMMON/WORLDC/    XB,YB,XT,YT,ILEVEL
      COMMON/DERDAT/    RCARS(40,13),NCARS,UWIND,UANGL,PATM,
     1                  TATM,UTRAIN,RCURV,ICARDR,ITRUCK,IAXLE,
     1                  CENTER,D1X,D1Y,D2X,D2Y,
     1                  USTR,UEND,DX,DY,IDTRK,ISPEED,CENX,CENY,
     1                  NUMTNK,DERAIL,DERTYP,ICAR,ATMOSP
      COMMON/TRKK/      ITRACK

      CALL FIXLN()
      CALL DWIN()
      CALL MOVTCA(1.,2.)
      CALL TEXT('/LOCATE POINT/')
      CALL MOVTCA(1.,4.)
      CALL TEXT('/OF DERAILMENT/')
      CALL DELTCU()
      CALL RESETW()

      IF ((D1X-D2X) .EQ. 0) GOTO 2000

      M = (D1Y-D2Y)/(D1X-D2X)
      X = D1X
      Y = D1Y

10    CALL MOVHCA(X,Y)
      CALL ORGLOC(X,Y)
      CALL DISXY(X,Y)
15    CALL READLO(X1,Y1,SWITCH)
      DO 33 KK = 1,1500
33    CONTINUE
      CALL READLO(NX,Y1,SWITCH)
      IF (SWITCH .EQ. 4 .OR. SWITCH .EQ. 132) GOTO 20
      IF (NX .EQ. X1) GOTO 15
      IF (NX .LT. D1X) THEN
          NX = D1X
      ELSEIF (NX .GT. D2X) THEN
          NX = D2X
      ENDIF
      Y = M*(NX-D1X) + D1Y
      X = NX
      GOTO 10

20    DY = M*(NX-D1X) + D1Y
      DX = NX
      CALL CLRXY()
      GOTO 900
******
******
```

```
2000  CONTINUE
      X = D1X
      Y = D1Y

100   CALL MOVHCA(X,Y)
      CALL ORGLOC(X,Y)
      CALL DISXY(X,Y)
150   CALL READLO(X1,Y1,SWITCH)
      DO 330 KK = 1,1500
330   CONTINUE
      CALL READLO(NX,Y1,SWITCH)
      IF (SWITCH .EQ. 4 .OR. SWITCH .EQ. 132) GOTO 200
      IF (NX .EQ. X1) GOTO 150
      Y = Y - (NX - X1)
      IF (Y .LT. MIN(D2Y,D1Y)) THEN
        Y = MIN(D2Y,D1Y)
      ELSEIF (Y .GT. MAX(D1Y,D2Y)) THEN
        Y = MAX(D1Y,D2Y)
      ENDIF
      GOTO 100

200   DY = Y + (NX - X1)
      IF (Y .LT. MIN(D2Y,D1Y)) THEN
        Y = MIN(D2Y,D1Y)
      ELSEIF (Y .GT. MAX(D1Y,D2Y)) THEN
        Y = MAX(D1Y,D2Y)
      ENDIF
      DX = D1X
      CALL CLRXY()

900   CALL DRICON()
      CALL RESETW()

10000 END

************************************************************************
************************************************************************

      SUBROUTINE DERTRK()

      INTEGER          SWITCH
      REAL             X1,Y1,X,Y
      REAL             WX1,WY1,WX2,WY2,RINC,AINC
      INTEGER          NBLUE,GREEN,LBLUE,RED,VIOLET,BROWN,
     1                 WHITE,GRAY,BBLUE,BGREEN,BBBLUE,BRED,
     1                 BVIOLT,YELLOW,BWHITE
      COMMON/COLORS/   NBLUE,GREEN,LBLUE,RED,VIOLET,BROWN,
     1                 WHITE,GRAY,BBLUE,BGREEN,BBBLUE,BRED,
     1                 BVIOLT,YELLOW,BWHITE
      COMMON/ARRAS/    INTARR(100,10),REARR(100,100,2),KOS
      COMMON/DERDAT/   RCARS(40,13),NCARS,UWIND,UANGL,PATM,
     1                 TATM,UTRAIN,RCURV,ICARDR,ITRUCK,IAXLE,
     1                   CENTER,D1X,D1Y,D2X,D2Y,
     1                   USTR,UEND,DX,DY,IDTRK,ISPEED,CENX,CENY,
     1                   NUMTNK,DERAIL,DERTYP,ICAR,ATMOSP
      COMMON/TRKK/       ITRACK
      COMMON/DIRC/       DIRECT
      COMMON/TRANS/      THETA
      COMMON/ROTATE/     BETA



      CALL FIXLN()
      CALL DWIN()
      CALL MOVTCA(1.,2.)
      CALL TEXT('/LOCATE POINT/')
      CALL MOVTCA(1.,4.)
      CALL TEXT('/OF DERAILMENT/')
      CALL DELTCU()
      CALL RESETW()

      X1 = CENX
      Y1 = CENY
      RINC = USTR
      AINC = .02
      X = RCURV*COS(RINC)
      Y = RCURV*SIN(RINC)

      PX = X+X1
10    CALL MOVHCA(X+X1,Y1-Y)
      CALL ORGLOC(X+X1,Y1-Y)
      CALL DISXY(X+X1,Y1-Y)
15    CALL READLO(RX,Y,SWITCH)
      DO 33 KK = 1,1500
33    CONTINUE
      CALL READLO(X,Y,SWITCH)
      IF (SWITCH .EQ. 4 .OR. SWITCH .EQ. 132) GOTO 20
      IF (RX .EQ. X) GOTO 15

      IF ((RX - X) .LT. 0) THEN
         RINC = RINC - AINC
      ELSEIF ((RX - X) .GT. 0) THEN
         RINC = RINC + AINC
      ENDIF
      IF (RINC .LT. USTR) THEN
         RINC = USTR
      ELSEIF (RINC .GT. UEND) THEN
         RINC = UEND
      ENDIF
      X = RCURV*COS(RINC)
      Y = RCURV*SIN(RINC)
      GOTO 10

20    CALL CLRXY()
```

```
      X = RCURV*COS(RINC)
      Y = RCURV*SIN(RINC)
      X = X + X1
      Y = Y1 - Y
      DDX = X - CENX
      IF (DDX .EQ. 0.) DDX = .0001
      THETA = RINC
      UANGL = RINC
      BETA = 1.57 - RINC
      IF (DIRECT .GT. 0.) THEN
         UANGL = 1.57 - UANGL
      ELSE
         UANGL = 4.7123 - UANGL
      ENDIF
      DX = X
      DY = Y

10000 CALL DRICON()
      END

************************************************************************
************************************************************************

      SUBROUTINE CALCAR(I)
      INTEGER          I

      REAL             TH,X1,Y1
      INTEGER          NBLUE,GREEN,LBLUE,RED,VIOLET,BROWN,
     1                 WHITE,GRAY,BBLUE,BGREEN,BBBLUE,BRED,
     1                 BVIOLT,YELLOW,BWHITE
      COMMON/COLORS/   NBLUE,GREEN,LBLUE,RED,VIOLET,BROWN,
     1                 WHITE,GRAY,BBLUE,BGREEN,BBBLUE,BRED,
     1                 BVIOLT,YELLOW,BWHITE
      COMMON/ARRAS/    INTARR(100,10),REARR(100,100,2),KOS
      COMMON/WORLDC/   XB,YB,XT,YT,ILEVEL
      COMMON/DERDAT/   RCARS(40,13),NCARS,UWIND,UANGL,PATM,
     1                 TATM,UTRAIN,RCURV,ICARDR,ITRUCK,IAXLE,
     1                 CENTER,D1X,D1Y,D2X,D2Y,
     1                 USTR,UEND,DX,DY,IDTRK,ISPEED,CENX,CENY,
     1                 NUMTNK,DERAIL,DERTYP,ICAR,ATMOSP
      COMMON/TRANS/    THETA
      COMMON/ROTATE/   BETA
      COMMON/DIRC/     DIRECT

      X1 = RCARS(I,6)
      Y1 = RCARS(I,7)

      IF (CENTER .EQ. 2.0) THEN
         IF (DIRECT .LT. 0.) THEN
            TH = BETA
         ELSE
```

```
              TH = -(BETA)
           ENDIF
           A =   COS(TH)
           B =   SIN(TH)
           C = -(SIN(TH))
           D =   COS(TH)
           TX = A*X1 + C*Y1
           TY = B*X1 + D*Y1

           X1 = CENX + RCURV*COS(THETA)
           Y1 = CENY - RCURV*SIN(THETA)
           IF (DIRECT .GT. 0.) THEN
              RCARS(I,6)  = TX + X1
              RCARS(I,7)  = Y1 - TY
           ELSE
              RCARS(I,6) = X1 - TX
              RCARS(I,7) = Y1 - TY
           ENDIF
        ELSE
           TH = BETA
           A =   COS(TH)
           B =   SIN(TH)
           C = -(SIN(TH))
           D =   COS(TH)
           TX = A*X1 + C*Y1
           TY = B*X1 + D*Y1
           X1 = DX
           Y1 = DY
           IF (DIRECT .GT. 0.) THEN
              RCARS(I,6)  = X1 + TX
              RCARS(I,7)  = Y1 - TY
           ELSE
              RCARS(I,6)  = X1 - TX
              RCARS(I,7)  = Y1 - TY
           ENDIF
        ENDIF

        IF (RCARS(I,6) .EQ. 0. .AND. RCARS(I,7) .EQ. 0.) THEN
           RCARS(I,6) = DX
           RCARS(I,7) = DY
        ENDIF
10000   END

************************************************************************
************************************************************************

        SUBROUTINE SETRKW(TYPE)
        INTEGER           TYPE
        COMMON/WORLDC/    XB,YB,XT,YT,ILEVEL

        IF     (TYPE .EQ. 5) THEN
           CALL SETCOL(0)
           CALL SETCOL(0)
           CALL SETLNS(3)
           CALL SETLNS(3)
           CALL SETLNW(3*ILEVEL)
           CALL SETLNW(3*ILEVEL)
        ELSEIF (TYPE .EQ. 6) THEN
           CALL SETCOL(0)
           CALL SETCOL(0)
           CALL SETLNS(3)
           CALL SETLNS(3)
           CALL SETLNW(5*ILEVEL)
           CALL SETLNW(5*ILEVEL)
        ENDIF
        END

************************************************************************
************************************************************************


        SUBROUTINE JOINED()
        INTEGER          TYPE,SWITCH
        REAL             WX1,WY1,WX2,WY2
        INTEGER          NUMCR
        INTEGER          NBLUE,GREEN,LBLUE,RED,VIOLET,BROWN,
     1                  WHITE,GRAY,BBLUE,BGREEN,BBBLUE,BRED,
     1                  BVIOLT,YELLOW,BWHITE
        COMMON/ARRAS/    INTARR(100,10),REARR(100,100,2),KOS
        COMMON/COLORS/   NBLUE,GREEN,LBLUE,RED,VIOLET,BROWN,
     1                  WHITE,GRAY,BBLUE,BGREEN,BBBLUE,BRED,
     1                  BVIOLT,YELLOW,BWHITE
        COMMON/DERDAT/   RCARS(40,13),NCARS,UWIND,UANGL,PATM,
     1                  TATM,UTRAIN,RCURV,ICARDR,ITRUCK,IAXLE,
     1                  CENTER,D1X,D1Y,D2X,D2Y,
     1                  USTR,UEND,DX,DY,IDTRK,ISPEED,CENX,CENY,
     1                  NUMTNK,DERAIL,DERTYP,ICAR,ATMOSP
        COMMON/TRKK/     ITRACK
        COMMON/DIRC/     DIRECT


        CALL FIXLN()
        CALL DWIN()
        CALL MOVTCA(1.,2.)
        CALL TEXT('/SELECT END/')
        CALL MOVTCA(1.,4.)
        CALL TEXT('/OF TRACK/')
        CALL MOVTCA(1.,8.)
        CALL TEXT('/TO JOIN;/')
        CALL MOVTCA(1.,10.)
        CALL TEXT('/1 or 2/')
        CALL DELTCU()

20      CALL PROMPT()
        READ(*,'(I10)',END=20,ERR=20)IEND
```

```
      IF (IEND .LT. 1 .OR. IEND .GT. 2) GOTO 20
      CALL CPROM()
      CALL DRICON()
      D1Y = ABS(D1Y)
      D2Y = ABS(D2Y)

      IF(IEND .EQ. 1) THEN
        REARR(KOS,1,1) = D1X
        REARR(KOS,1,2) = D1Y
      ELSE
        REARR(KOS,1,1) = D2X
        REARR(KOS,1,2) = D2Y
      ENDIF
      INTARR(KOS,1) = IDTRK+4
      INTARR(KOS,2) = 1

      CALL RESETW()
      CALL FIXLN()
      CALL DELHCU()
      CALL ORGLOC(REARR(KOS,1,1),REARR(KOS,1,2))
      CALL MOVABS(REARR(KOS,1,1),REARR(KOS,1,2))

30    CALL READLO(X1,Y1,SWITCH)
      CALL DISXY(X1,Y1)
      CALL RLNABS(X1,Y1)
      IF (SWITCH .EQ. 4 .OR. SWITCH .EQ. 132) THEN
         CALL CLRXY()
         CALL FIXLN()
         CALL WAIT4()
         CALL LNABS(X1,Y1)
         INTARR(KOS,2) = INTARR(KOS,2) + 1
         REARR(KOS,INTARR(KOS,2),1) = X1
         REARR(KOS,INTARR(KOS,2),2) = Y1
         CALL SETXOR(1)
      ELSEIF( SWITCH .EQ. 2 .OR. SWITCH .EQ. 130) THEN
             CALL CLRXY()
             CALL DELLN()
             CALL WAIT2()
             CALL SETXOR(0)
             IF (INTARR(KOS,2) .LT. 2) THEN
                CONTINUE
             ELSE
                KOS = KOS + 1
             ENDIF
             GOTO 10000
      ENDIF
         GOTO 30
10000 CALL SETXOR(0)
      CALL SUBRDR()
      END

****************************************************************************
```

```
***************************************************************************
```

```
************************************************************************
************************************************************************


      SUBROUTINE REPUF(I)
      INTEGER          I

      COMMON/DERDAT/   RCARS(40,13),NCARS,UWIND,UANGL,PATM,
     1                 TATM,UTRAIN,RCURV,ICARDR,ITRUCK,IAXLE,
     1                 CENTER,D1X,D1Y,D2X,D2Y,
     1                 USTR,UEND,DX,DY,IDTRK,ISPEED,CENX,CENY,
     1                 NUMTNK,DERAIL,DERTYP,ICAR,ATMOSP
      COMMON/RESULT1/  IBLAST(8),BLAST(8,7),IFLARE(8),FLAR(8,3),
     1                 IPUFF(8),PUF(8,3),ISP(8),SP(8,6),
     1                 IPLUM(8),PLM(8,2),PLMM(300,2),
     1                 IFIRE(8),FIRE(8,4),RUPDAT(8,5),IPLM(8),IPUF(8)
      COMMON/GLBTM/    IGLBTM
      COMMON/DET/      IDETL(7)

      IF (IPUF(I) .EQ. 1) THEN
         READ(20+I,*)(PUF(I,KK),KK=2,3)
      ENDIF
10000 END

************************************************************************
************************************************************************

      SUBROUTINE REPLM(I)
      INTEGER          I


      COMMON/DERDAT/   RCARS(40,13),NCARS,UWIND,UANGL,PATM,
     1                 TATM,UTRAIN,RCURV,ICARDR,ITRUCK,IAXLE,
     1                 CENTER,D1X,D1Y,D2X,D2Y,
     1                 USTR,UEND,DX,DY,IDTRK,ISPEED,CENX,CENY,
     1                 NUMTNK,DERAIL,DERTYP,ICAR,ATMOSP
      COMMON/RESULT1/  IBLAST(8),BLAST(8,7),IFLARE(8),FLAR(8,3),
     1                 IPUFF(8),PUF(8,3),ISP(8),SP(8,6),
     1                 IPLUM(8),PLM(8,2),PLMM(300,2),
     1                 IFIRE(8),FIRE(8,4),RUPDAT(8,5),IPLM(8),IPUF(8)
      COMMON/DET/      IDETL(7)
      COMMON/GLBTM/    IGLBTM
      IF (IPLUM(I) .EQ. 1) THEN
         DO 200 K = 1,300
            IREC = K + (IGLBTM) * 300
            READ(I+20,REC=IREC)(PLMM(K,KK),KK=1,2)
200      CONTINUE
      ENDIF
10000 END


************************************************************************
```

```
************************************************************************
      SUBROUTINE DFIRE(I)
      INTEGER         I

      INTEGER            NBLUE,GREEN,LBLUE,RED,VIOLET,BROWN,
     1                   WHITE,GRAY,BBLUE,BGREEN,BBBLUE,BRED,
     1                   BVIOLT,YELLOW,BWHITE
      COMMON/COLORS/     NBLUE,GREEN,LBLUE,RED,VIOLET,BROWN,
     1                   WHITE,GRAY,BBLUE,BGREEN,BBBLUE,BRED,
     1                   BVIOLT,YELLOW,BWHITE
      COMMON/WORLDC/     XB,YB,XT,YT,ILEVEL
      COMMON/DERDAT/     RCARS(40,13),NCARS,UWIND,UANGL,PATM,
     1                   TATM,UTRAIN,RCURV,ICARDR,ITRUCK,IAXLE,
     1                   CENTER,D1X,D1Y,D2X,D2Y,
     1                   USTR,UEND,DX,DY,IDTRK,ISPEED,CENX,CENY,
     1                   NUMTNK,DERAIL,DERTYP,ICAR,ATMOSP
      COMMON/RESULT1/    IBLAST(8),BLAST(8,7),IFLARE(8),FLAR(8,3),
     1                   IPUFF(8),PUF(8,3),ISP(8),SP(8,6),
     1                   IPLUM(8),PLM(8,2),PLMM(300,2),
     1                   IFIRE(8),FIRE(8,4),RUPDAT(8,5),IPLM(8),IPUF(8)
      COMMON/DET/        IDETL(7)
      COMMON/GLBTM/      IGLBTM


      IF (IFIRE(ICAR) .EQ. 0) GOTO 10000
      TH = RCARS(ICAR,8)
      A =   COS(TH)
      B =   SIN(TH)
      C = -(SIN(TH))
      D =   COS(TH)
      TX = A*(-2.0) + C*1.0
      TY = B*(-2.0) + D*1.0
      TX = RCARS(I,6) + TX
      TY = RCARS(I,7) - TY
      CALL SETLEV()
      IF ((ILEVEL .EQ. 3) .AND. (TX .GT. XB .AND. TX .LT. XT
     1     .AND. TY .GT. YB .AND. TY .LT. YT)) THEN
         CALL SETTEX(1,1,0,0)
         CALL SETTCL(BRED,BGREEN)
         CALL MOVTCA(TX,TY)
         CALL TEXT('/F/')
         CALL DELTCU()
      ELSE
         CALL SETCOL(BVIOLT)
         CALL SETCOL(BVIOLT)
      ENDIF

10000 END

************************************************************************
************************************************************************
```

```
      SUBROUTINE DSPLL(I)
      INTEGER              I
      INTEGER           NBLUE,GREEN,LBLUE,RED,VIOLET,BROWN,
     1                  WHITE,GRAY,BBLUE,BGREEN,BBBLUE,BRED,
     1                  BVIOLT,YELLOW,BWHITE
      COMMON/COLORS/    NBLUE,GREEN,LBLUE,RED,VIOLET,BROWN,
     1                  WHITE,GRAY,BBLUE,BGREEN,BBBLUE,BRED,
     1                  BVIOLT,YELLOW,BWHITE
      COMMON/DERDAT/    RCARS(40,13),NCARS,UWIND,UANGL,PATM,
     1                  TATM,UTRAIN,RCURV,ICARDR,ITRUCK,IAXLE,
     1                  CENTER,D1X,D1Y,D2X,D2Y,
     1                  USTR,UEND,DX,DY,IDTRK,ISPEED,CENX,CENY,
     1                  NUMTNK,DERAIL,DERTYP,ICAR,ATMOSP
      COMMON/RESULT1/   IBLAST(8),BLAST(8,7),IFLARE(8),FLAR(8,3),
     1                  IPUFF(8),PUF(8,3),ISP(8),SP(8,6),
     1                  IPLUM(8),PLM(8,2),PLMM(300,2),
     1                  IFIRE(8),FIRE(8,4),RUPDAT(8,5),IPLM(8),IPUF(8)
      COMMON/GLBTM/     IGLBTM
      COMMON/DET/       IDETAL(7)

      IF (ISP(ICAR) .EQ. 0) GOTO 10000
      CALL MOVTCA(RCARS(I,6),RCARS(I,7))
      CALL SETTCL(BRED,BGREEN)
      CALL SETTEX(1,1,0,0)
      CALL TEXT('/F/')
      CALL SETLNW(1)
      CALL SETCOL(RED)
      CALL SETCOL(RED)
      X = RCARS(I,6)
      Y = RCARS(I,7)
      CALL MOVABS(X,Y)
      CALL CIR(SP(ICAR,2)/2.0)
10000 END

************************************************************************
************************************************************************

      SUBROUTINE DBLAST(I)
      INTEGER              I
      LOGICAL           BLT,EXP
      INTEGER           NBLUE,GREEN,LBLUE,RED,VIOLET,BROWN,
     1                  WHITE,GRAY,BBLUE,BGREEN,BBBLUE,BRED,
     1                  BVIOLT,YELLOW,BWHITE
      COMMON/COLORS/    NBLUE,GREEN,LBLUE,RED,VIOLET,BROWN,
     1                  WHITE,GRAY,BBLUE,BGREEN,BBBLUE,BRED,
     1                  BVIOLT,YELLOW,BWHITE
      COMMON/DERDAT/    RCARS(40,13),NCARS,UWIND,UANGL,PATM,
     1                  TATM,UTRAIN,RCURV,ICARDR,ITRUCK,IAXLE,
     1                  CENTER,D1X,D1Y,D2X,D2Y,
     1                  USTR,UEND,DX,DY,IDTRK,ISPEED,CENX,CENY,
     1                  NUMTNK,DERAIL,DERTYP,ICAR,ATMOSP
```

```
      COMMON/WORLDC/   XB,YB,XT,YT,ILEVEL
      COMMON/RESULT1/  IBLAST(8),BLAST(8,7),IFLARE(8),FLAR(8,3),
     1                 IPUFF(8),PUF(8,3),ISP(8),SP(8,6),
     1                 IPLUM(8),PLM(8,2),PLMM(300,2),
     1                 IFIRE(8),FIRE(8,4),RUPDAT(8,5),IPLM(8),IPUF(8)
      COMMON/GLBTM/    IGLBTM
      COMMON/DET/      IDETAL(7)

      BLT = .TRUE.
      EXP = .TRUE.
      IF (IDETAL(1) .EQ. 0) BLT = .FALSE.
      IF (IDETAL(2) .EQ. 0) EXP = .FALSE.

      CX = RCARS(I,6)
      CY = RCARS(I,7)
      ANG = 0.
      DO 100 J = 1,10
         CALL MOVABS(CX,CY)
         CALL SETCOL(RED)
         CALL SETLNS(1)
         CALL SETLNW(1)
            IF (BLT)  CALL ARC(BLAST(ICAR,2),ANG,ANG+.528)
         CALL SETCOL(VOILET)
         CALL SETCOL(VIOLET)
            IF (EXP) CALL ARC(BLAST(ICAR,5),ANG,ANG+.528)
         ANG = ANG + .628
100   CONTINUE


      ANG = 0.
      DO 200 J = 1,20
         CALL MOVABS(CX,CY)
         CALL SETCOL(RED)
         CALL SETLNS(1)
         CALL SETLNW(1)
            IF (BLT) CALL ARC(BLAST(ICAR,3),ANG,ANG+.214)
         CALL SETCOL(VOILET)
         CALL SETCOL(VIOLET)
            IF (EXP) CALL ARC(BLAST(ICAR,6),ANG,ANG+.214)
         ANG = ANG + .314
200   CONTINUE

      ANG = 0.
      DO 300 J = 1,30
         CALL MOVABS(CX,CY)
         CALL SETCOL(RED)
         CALL SETLNS(1)
         CALL SETLNW(1)
            IF (BLT) CALL ARC(BLAST(ICAR,4),ANG,ANG+.109)
         CALL SETCOL(VOILET)
         CALL SETCOL(VIOLET)
            IF (EXP) CALL ARC(BLAST(ICAR,7),ANG,ANG+.109)
```

```
           ANG = ANG + .209
300     CONTINUE
10000   END

************************************************************************
************************************************************************

        SUBROUTINE DPUFF(I)
        INTEGER         I,UNIT,STRT
        REAL            X1,Y1,X2,Y2,X3,Y3,X4,Y4
        INTEGER         NBLUE,GREEN,LBLUE,RED,VIOLET,BROWN,
     1                  WHITE,GRAY,BBLUE,BGREEN,BBBLUE,BRED,
     1                  BVIOLT,YELLOW,BWHITE
        COMMON/COLORS/  NBLUE,GREEN,LBLUE,RED,VIOLET,BROWN,
     1                  WHITE,GRAY,BBLUE,BGREEN,BBBLUE,BRED,
     1                  BVIOLT,YELLOW,BWHITE
        COMMON/ARRAS/   INTARR(100,10),REARR(100,100,2),KOS
        COMMON/WORLDC/  XB,YB,XT,YT,ILEVEL
        COMMON/DERDAT/  RCARS(40,13),NCARS,UWIND,UANGL,PATM,
     1                  TATM,UTRAIN,RCURV,ICARDR,ITRUCK,IAXLE,
     1                  CENTER,D1X,D1Y,D2X,D2Y,
     1                  USTR,UEND,DX,DY,IDTRK,ISPEED,CENX,CENY,
     1                  NUMTNK,DERAIL,DERTYP,ICAR,ATMOSP
        COMMON/RESULT1/ IBLAST(8),BLAST(8,7),IFLARE(8),FLAR(8,3),
     1                  IPUFF(8),PUF(8,3),ISP(8),SP(8,6),
     1                  IPLUM(8),PLM(8,2),PLMM(300,2),
     1                  IFIRE(8),FIRE(8,4),RUPDAT(8,5),IPLM(8),IPUF(8)
        COMMON/GLBTM/   IGLBTM
        COMMON/DET/     IDETAL(7)


        IF (IPUF(ICAR) .EQ. 0.) GOTO 10000
        X1 = RCARS(I,6)
        Y1 = RCARS(I,7)
        UW = UWIND
        IF (UWIND .GT. 3.14159) UW = UWIND - 6.2832
        X2 = PUF(ICAR,2) * COS(UW) + X1
        Y2 = Y1 - PUF(ICAR,2) * SIN(UW)
        CALL SETCOL(YELLOW)
        CALL SETCOL(YELLOW)
        CALL SETHAT(4)
        CALL SETHAT(4)
        CALL MOVABS(X2,Y2)
        CALL FCIR(PUF(ICAR,3))
        START = 5.28 - (1.57 - UW)
        CALL ARC(PUF(ICAR,3),START,START+3.14)
        CALL INQARC(X3,Y3,X4,Y4)
        CALL MOVABS(X1,Y1)
        CALL SETLNS(1)
        CALL SETLNS(1)
        CALL SETLNW(1)
        CALL LNABS(X3,Y3)
```

```
      CALL MOVABS(X1,Y1)
      CALL LNABS(X4,Y4)
10000 END

*******************************************************************
*******************************************************************


      SUBROUTINE DPLM(I)
      INTEGER          I,UNIT,STRT
      REAL             XD,RD,NX,NY
      INTEGER          NBLUE,GREEN,LBLUE,RED,VIOLET,BROWN,
     1                 WHITE,GRAY,BBLUE,BGREEN,BBBLUE,BRED,
     1                 BVIOLT,YELLOW,BWHITE
      COMMON/COLORS/   NBLUE,GREEN,LBLUE,RED,VIOLET,BROWN,
     1                 WHITE,GRAY,BBLUE,BGREEN,BBBLUE,BRED,
     1                 BVIOLT,YELLOW,BWHITE
      COMMON/ARRAS/    INTARR(100,10),REARR(100,100,2),KOS
      COMMON/WORLDC/   XB,YB,XT,YT,ILEVEL
      COMMON/DERDAT/   RCARS(40,13),NCARS,UWIND,UANGL,PATM,
     1                 TATM,UTRAIN,RCURV,ICARDR,ITRUCK,IAXLE,
     1                 CENTER,D1X,D1Y,D2X,D2Y,
     1                 USTR,UEND,DX,DY,IDTRK,ISPEED,CENX,CENY,
     1                 NUMTNK,DERAIL,DERTYP,ICAR,ATMOSP
      COMMON/RESULT1/  IBLAST(8),BLAST(8,7),IFLARE(8),FLAR(8,3),
     1                 IPUFF(8),PUF(8,3),ISP(8),SP(8,6),
     1                 IPLUM(8),PLM(8,2),PLMM(300,2),
     1                 IFIRE(8),FIRE(8,4),RUPDAT(8,5),IPLM(8),IPUF(8)
      COMMON/GLBTM/    IGLBTM
      COMMON/DET/      IDETAL(7)

      IF (IPLUM(ICAR) .EQ. 0.) GOTO 10000
      X1 = RCARS(I,6)
      Y1 = RCARS(I,7)
      UW = UWIND
      IF (UWIND .GT. 3.14159) UW = UWIND - 6.2832
      DO 100 J = 1,300
         NX = PLMM(J,1) * COS(UW) + X1
         NY = Y1 - PLMM(J,1) * SIN(UW)
         IF (NX .GT. XB .AND. NX .LT. XT .AND. NY .GT. YB
     1        .AND. NY .LT. YT) THEN
            CALL SETCOL(RED)
            CALL SETCOL(RED)
            CALL SETHAT(1)
            CALL SETHAT(1)
            CALL MOVABS(NX,NY)
            CALL FCIR(PLMM(J,2))
         ENDIF
100   CONTINUE
10000 END
```

```
**********************************************************************
**********************************************************************

      SUBROUTINE DFLAR(I)
      INTEGER          I
      REAL             X1,Y1,X2,Y2,X3,Y3
      INTEGER          NBLUE,GREEN,LBLUE,RED,VIOLET,BROWN,
     1                 WHITE,GRAY,BBLUE,BGREEN,BBBLUE,BRED,
     1                 BVIOLT,YELLOW,BWHITE
      COMMON/COLORS/   NBLUE,GREEN,LBLUE,RED,VIOLET,BROWN,
     1                 WHITE,GRAY,BBLUE,BGREEN,BBBLUE,BRED,
     1                 BVIOLT,YELLOW,BWHITE
      COMMON/DERDAT/   RCARS(40,13),NCARS,UWIND,UANGL,PATM,
     1                 TATM,UTRAIN,RCURV,ICARDR,ITRUCK,IAXLE,
     1                 CENTER,D1X,D1Y,D2X,D2Y,
     1                 USTR,UEND,DX,DY,IDTRK,ISPEED,CENX,CENY,
     1                 NUMTNK,DERAIL,DERTYP,ICAR,ATMOSP
      COMMON/WORLDC/   XB,YB,XT,YT,ILEVEL
      COMMON/RESULT1/  IBLAST(8),BLAST(8,7),IFLARE(8),FLAR(8,3),
     1                 IPUFF(8),PUF(8,3),ISP(8),SP(8,6),
     1                 IPLUM(8),PLM(8,2),PLMM(300,2),
     1                 IFIRE(8),FIRE(8,4),RUPDAT(8,5),IPLM(8),IPUF(8)
      COMMON/GLBTM/    IGLBTM
      COMMON/DET/      IDETAL(7)

      IF (IFLARE(ICAR) .EQ. 0) GOTO 10000
      CALL SETCOL(BRED)
      CALL SETCOL(BRED)
      CALL SETLNS(3)
      CALL SETLNS(3)
      CALL SETLNW(1)
      CALL SETLNW(1)
      X1 = RCARS(I,6)
      Y1 = RCARS(I,7)
      ANG = RCARS(I,8) - UANGL
      IF (RCARS(I,9) .LT. 0.)  THEN
         ANG = ANG - 1.57
      ELSE
         ANG = ANG + 1.57
      ENDIF
      PX = FLAR(ICAR,3) * COS(ANG) + X1
      PY = Y1 - FLAR(ICAR,3) * SIN(ANG)
      CALL SETLNW(3)
      CALL LNABS(PX,PY)
      CALL SETTEX(1,1,0,0)
      CALL SETTCL(BRED,BGREEN)
      CALL MOVTCA(PX,PY)
      CALL TEXT('/F/')
      CALL SETLNW(1)
10000 END
```

```
************************************************************************
************************************************************************
```

```
      PROGRAM TT
      INTEGER           NBLUE,GREEN,LBLUE,RED,VIOLET,BROWN,
     1                  WHITE,GRAY,BBLUE,BGREEN,BBBLUE,BRED,
     1                  BVIOLT,YELLOW,BWHITE
      LOGICAL           OUTFLG
      COMMON/COLORS/    NBLUE,GREEN,LBLUE,RED,VIOLET,BROWN,
     1                  WHITE,GRAY,BBLUE,BGREEN,BBBLUE,BRED,
     1                  BVIOLT,YELLOW,BWHITE
      COMMON/ARRAS/     INTARR(100,10),REARR(100,100,2),KOS
      COMMON/WORLDC/    XB,YB,XT,YT,ILEVEL
      COMMON/DERDAT/    RCARS(40,13),NCARS,UWIND,UANGL,PATM,
     1                  TATM,UTRAIN,RCURV,ICARDR,ITRUCK,IAXLE,
     1                  CENTER,D1X,D1Y,D2X,D2Y,
     1                  USTR,UEND,DX,DY,IDTRK,ISPEED,CENX,CENY,
     1                  NUMTNK,DERAIL,DERTYP,ICAR,ATMOSP
      COMMON/RESULT1/   IBLAST(8),BLAST(8,7),IFLARE(8),FLAR(8,3),
     1                  IPUFF(8),PUF(8,3),ISP(8),SP(8,6),
     1                  IPLUM(8),PLM(8,2),PLMM(300,2),
     1                  IFIRE(8),FIRE(8,4),RUPDAT(8,5),IPLM(8),IPUF(8)
      COMMON/GLBTM/     IGLBTM
      COMMON/DET/       IDETAL(7)
      COMMON/MODE/      IMODE
      COMMON/OUTFL/     OUTFLG
      COMMON/WINDS/     WSPEED
      COMMON/BLK/       IBLACK

      CALL INIPST()
      CALL PPOST()
10000 WRITE(*,*)' 11;10HPROGRAM   1mEXIT0m'
      END

*************************************************************************
*************************************************************************
```

```
        PROGRAM MAIN
        CALL PROCC()
10000   END

        SUBROUTINE PROCC()

10      OPEN(21,FILE='TCAR1.PUF',STATUS='OLD')
        READ(21,*,END=20)RNUM
        IF (RNUM .NE. -1) THEN
           CLOSE(21)
           OPEN(21,FILE='TCAR1.PUF',STATUS='OLD')
           OPEN(61,FILE='PUFF1.ALT',STATUS='OLD',ACCESS='DIRECT',
     1             RECL=8)
           CALL PROCCF(21,61)
           CLOSE(21)
           CLOSE(61)
        ENDIF


20      OPEN(22,FILE='TCAR2.PUF',STATUS='OLD')
        READ(22,*,END=30)RNUM
        IF (RNUM .NE. -1) THEN
           CLOSE(22)
           OPEN(22,FILE='TCAR2.PUF',STATUS='OLD')
           OPEN(62,FILE='PUFF2.ALT',STATUS='OLD',ACCESS='DIRECT',
     1             RECL=8)
           CALL PROCCF(22,62)
           CLOSE(22)
           CLOSE(62)
        ENDIF


30      OPEN(23,FILE='TCAR3.PUF',STATUS='OLD')
        READ(23,*,END=40)RNUM
        IF (RNUM .NE. -1) THEN
           CLOSE(23)
           OPEN(23,FILE='TCAR3.PUF',STATUS='OLD')
           OPEN(63,FILE='PUFF3.ALT',STATUS='OLD',ACCESS='DIRECT',
     1             RECL=8)
           CALL PROCCF(23,63)
           CLOSE(23)
           CLOSE(63)
        ENDIF


40      OPEN(24,FILE='TCAR4.PUF',STATUS='OLD')
        READ(24,*,END=50)RNUM
        IF (RNUM .NE. -1) THEN
           CLOSE(24)
           OPEN(24,FILE='TCAR4.PUF',STATUS='OLD')
           OPEN(64,FILE='PUFF4.ALT',STATUS='OLD',ACCESS='DIRECT',
     1             RECL=8)
```

```
            CALL PROCCF(24,64)
            CLOSE(24)
            CLOSE(64)
         ENDIF

 50      OPEN(25,FILE='TCAR5.PUF',STATUS='OLD')
         READ(25,*,END=60)RNUM
         IF (RNUM .NE. -1) THEN
            CLOSE(25)
            OPEN(25,FILE='TCAR5.PUF',STATUS='OLD')
            OPEN(65,FILE='PUFF5.ALT',STATUS='OLD',ACCESS='DIRECT',
     1               RECL=8)
            CALL PROCCF(25,65)
            CLOSE(25)
            CLOSE(65)
         ENDIF


 60      OPEN(26,FILE='TCAR6.PUF',STATUS='OLD')
         READ(26,*,END=70)RNUM
         IF (RNUM .NE. -1) THEN
            CLOSE(26)
            OPEN(26,FILE='TCAR6.PUF',STATUS='OLD')
            OPEN(66,FILE='PUFF6.ALT',STATUS='OLD',ACCESS='DIRECT',
     1               RECL=8)
            CALL PROCCF(26,66)
            CLOSE(26)
            CLOSE(66)
         ENDIF


 70      OPEN(27,FILE='TCAR7.PUF',STATUS='OLD')
         READ(27,*,END=80)RNUM
         IF (RNUM .NE. -1) THEN
            CLOSE(27)
            OPEN(27,FILE='TCAR7.PUF',STATUS='OLD')
            OPEN(67,FILE='PUFF7.ALT',STATUS='OLD',ACCESS='DIRECT',
     1               RECL=8)
            CALL PROCCF(27,67)
            CLOSE(27)
            CLOSE(67)
         ENDIF



 80      OPEN(28,FILE='TCAR8.PUF',STATUS='OLD')
         READ(28,*,END=10000)RNUM
         IF (RNUM .NE. -1) THEN
            CLOSE(28)
            OPEN(28,FILE='TCAR8.PUF',STATUS='OLD')
            OPEN(68,FILE='PUFF8.ALT',STATUS='OLD',ACCESS='DIRECT',
     1               RECL=8)
```

```
           CALL PROCCF(28,68)
           CLOSE(28)
           CLOSE(68)
        ENDIF

10000   END

**********************************************************************
**********************************************************************


        SUBROUTINE PROCCF(INIT1,INIT2)
        INTEGER                  INIT1,INIT2
        REAL                     PARR(150,300,2),TIME

        ICOUNT = 1
        DO 100 I = 1,150
           READ(INIT1,*,END=101)TIME
           WRITE(*,*)TIME
           DO 50 K = 1,300
              READ(INIT1,*)(PARR(I,K,KK),KK=1,2)
50         CONTINUE
           ICOUNT = ICOUNT + 1
100     CONTINUE
101     CONTINUE

        DO 200 I = 1,300
           ISPOT = I
           DO 150 K = 1,I
              IREC = K +(I-1)*300
              IF (K .GT. (ICOUNT+1)) GOTO 151
              WRITE(INIT2,REC=IREC)(PARR(K,ISPOT,KK),KK=1,2)
              ISPOT = ISPOT - 1
150        CONTINUE
151        CONTINUE
200     CONTINUE

        IF (ICOUNT .LT. 150) GOTO 10000
        ICNT = 0

        DO 300 I = 1,149
           READ(INIT1,*,END=301)TIME
           DO 250 K = 1,300
              READ(INIT1,*)(PARR(I,K,KK),KK=1,2)
250        CONTINUE
300     CONTINUE
301     CONTINUE

        DO 320 I = 1,150
           DO 310 K = 1,300
              PARR(I,K,1) = 0.
              PARR(I,K,2) = 0.
```

```
310        CONTINUE
320     CONTINUE

        DO 400 I = 1,150
           ISPOT = I
           DO 350 K = 1,I
              IREC= (I+ICOUNT-1) * 300 + J
              WRITE(INIT2,REC=IREC)(PARR(K,ISPOT,KK),KK=1,2)
              ISPOT = ISPOT - 1
350        CONTINUE
400     CONTINUE
10000   END
```

```
        SUBROUTINE NEXTST()
        REAL             TIME
        LOGICAL          OUTFLG

        COMMON/DERDAT/   RCARS(40,13),NCARS,UWIND,UANGL,PATM,
     1                   TATM,UTRAIN,RCURV,ICARDR,ITRUCK,IAXLE,
     1                   CENTER,D1X,D1Y,D2X,D2Y,
     1                   USTR,UEND,DX,DY,IDTRK,ISPEED,CENX,CENY,
     1                   NUMTNK,DERAIL,DERTYP,ICAR,ATMOSP
        COMMON/WORLDC/   XB,YB,XT,YT,ILEVEL
        COMMON/RESULT1/  IBLAST(8),BLAST(8,7),IFLARE(8),FLAR(8,3),
     1                   IPUFF(8),PUF(8,3),ISP(8),SP(8,6),
     1                   IPLUM(8),PLM(8,2),PLMM(300,2),
     1                   IFIRE(8),FIRE(8,4),RUPDAT(8,5),IPLM(8),IPUF(8)
        COMMON/GLTM/     IGLBTM
        COMMON/DET/      IDETAL(7)
        COMMON/OUTFL/    OUTFLG
        COMMON/TANK/     ITCAR(40)

***   READ NEXT STEP IN TIME AND DISPLAY EVENTS

        READ(5,*,END=900,ERR=900)TIME
        ICAR = 0
        DO 50 I = 1,NCARS
           IF (RCARS(I,1) .EQ. 2.) THEN
              ICAR = ICAR + 1
      READ(5,*,END=900,ERR=900)IC,IFIRE(ICAR),(FIRE(ICAR,KK),KK=1,4)
      READ(5,*,END=900,ERR=900)(RUPDAT(ICAR,KK),KK=1,5)
      READ(5,*,END=900,ERR=900)ISP(ICAR),(SP(ICAR,KK),KK=1,6)
      READ(5,*,END=900,ERR=900)IFLARE(ICAR),(FLAR(ICAR,KK),KK=1,3)
      READ(5,*,END=900,ERR=900)IBLAST(ICAR),(BLAST(ICAR,KK),KK=1,7)
      READ(5,*,END=900,ERR=900)IPLUM(ICAR),(PLM(ICAR,KK),KK=1,2)
      READ(5,*,END=900,ERR=900)IPUF(ICAR),PUF(ICAR,1)
              IF (RUPDAT(ICAR,1) .GT. 0.) THEN
                 IF (RUPDAT(ICAR,2) .GT. 0.) THEN
                     RCARS(I,10) = RUPDAT(ICAR,1)
                     RCARS(I,11) = RUPDAT(ICAR,3)
                     RCARS(I,12) = RUPDAT(ICAR,4)
                     RCARS(I,13) = RUPDAT(ICAR,5)
                 ENDIF
              ENDIF
              CALL REPLM(ICAR)
              CALL REPUF(ICAR)
              CALL RECOD(I)
           ENDIF
50      CONTINUE

        CALL SUBRDR()
        ICAR = 0
        GOTO 10000
```

```
900     CONTINUE
        OUTFLG = .TRUE.
10000   END

**********************************************************************
**********************************************************************


        SUBROUTINE RECOD(I)
        INTEGER          I


        COMMON/DERDAT/   RCARS(40,13),NCARS,UWIND,UANGL,PATM,
     1                   TATM,UTRAIN,RCURV,ICARDR,ITRUCK,IAXLE,
     1                   CENTER,D1X,D1Y,D2X,D2Y,
     1                   USTR,UEND,DX,DY,IDTRK,ISPEED,CENX,CENY,
     1                   NUMTNK,DERAIL,DERTYP,ICAR,ATMOSP
        COMMON/WORLDC/   XB,YB,XT,YT,ILEVEL
        COMMON/RESULT1/  IBLAST(8),BLAST(8,7),IFLARE(8),FLAR(8,3),
     1                   IPUFF(8),PUF(8,3),ISP(8),SP(8,6),
     1                   IPLUM(8),PLM(8,2),PLMM(300,2),
     1                   IFIRE(8),FIRE(8,4),RUPDAT(8,5),IPLM(8),IPUF(8)
        COMMON/GLTM/     GLBTM
        COMMON/DET/      IDETAL(7)
        COMMON/OUTFL/    OUTFLG
        COMMON/TANK/     ITCAR(40)
        COMMON/MREC/     RSPL(8,8),RBLAST(8,9),RRUPT(8,6),RFIRE(8,6),
     1                   RFLAR(8,5),RPLUM(8,5),RPUFF(8,5)
        COMMON/RUPT/     RPT(8)


        IF (RPT(ICAR) .EQ. 1.0 .AND. RRUPT(ICAR,1) .NE. 1.0) THEN
           RRUPT(ICAR,1) = 1.0
           RRUPT(ICAR,2) = RCARS(I,10)
           RRUPT(ICAR,3) = 0.
           RRUPT(ICAR,4) = RCARS(I,11)
           RRUPT(ICAR,5) = RCARS(I,12)
           RRUPT(ICAR,6) = RCARS(1,13)
        ELSEIF (RUPDAT(ICAR,1) .GT. 0. .AND. RPT(ICAR) .NE. 1.0) THEN
           IF (RRUPT(ICAR,1) .NE. 2.0) THEN
              RRUPT(ICAR,1) = 2.0
              RRUPT(ICAR,2) = RUPDAT(ICAR,1)
              RRUPT(ICAR,3) = RUPDAT(ICAR,2)
              RRUPT(ICAR,4) = RUPDAT(ICAR,3)
              RRUPT(ICAR,5) = RUPDAT(ICAR,4)
              RRUPT(ICAR,6) = RUPDAT(ICAR,5)
           ENDIF
        ENDIF

        IF (ISP(ICAR) .GE. 1) THEN
           IF (RSPL(ICAR,1) .EQ. 0.) THEN
```

```
              RSPL(ICAR,1) = 1.0
              RSPL(ICAR,2) = REAL(ISP(ICAR))
              RSPL(ICAR,3) = SP(ICAR,1)
           ENDIF
           RSPL(ICAR,4) = MAX(SP(ICAR,2),RSPL(ICAR,4))
           RSPL(ICAR,8) = SP(ICAR,6)
        ENDIF


        IF (IFIRE(ICAR) .GT. 0) THEN
           IF (RFIRE(ICAR,1) .EQ. 0.) THEN
              RFIRE(ICAR,1) = 1.0
              RFIRE(ICAR,2) = REAL(IFIRE(ICAR))
              RFIRE(ICAR,3) = FIRE(ICAR,2)
              RFIRE(ICAR,4) = FIRE(ICAR,3)
           ENDIF
        ENDIF

        IF (IFLARE(ICAR) .GT. 0) THEN
           IF (RFLAR(ICAR,1) .EQ. 0) THEN
              RFLAR(ICAR,1) = 1.0
              RFLAR(ICAR,2) = REAL(IFLARE(ICAR))
              RFLAR(ICAR,3) = FLAR(ICAR,1)
              RFLAR(ICAR,4) = FLAR(ICAR,2)
           ENDIF
           RFLAR(ICAR,5) = MAX(RFLAR(ICAR,5),FLAR(ICAR,3))
        ENDIF

        IF (IBLAST(ICAR) .GT. 0) THEN
           IF (RBLAST(ICAR,1) .EQ. 0.) THEN
              RBLAST(ICAR,1) = 1.0
              RBLAST(ICAR,2) = REAL(IBLAST(ICAR))
              RBLAST(ICAR,3) = BLAST(ICAR,1)
           ENDIF
           RBLAST(ICAR,4) = MAX(RBLAST(ICAR,4),BLAST(ICAR,2))
           RBLAST(ICAR,5) = MAX(RBLAST(ICAR,5),BLAST(ICAR,3))
           RBLAST(ICAR,6) = MAX(RBLAST(ICAR,6),BLAST(ICAR,4))
           RBLAST(ICAR,7) = MAX(RBLAST(ICAR,7),BLAST(ICAR,5))
           RBLAST(ICAR,8) = MAX(RBLAST(ICAR,8),BLAST(ICAR,6))
           RBLAST(ICAR,9) = MAX(RBLAST(ICAR,9),BLAST(ICAR,7))
        ENDIF

        IF (IPUF(ICAR) .GT. 0) THEN
           IF (RPUFF(ICAR,1) .EQ. 0.) THEN
              RPUFF(ICAR,1) = 1.0
              RPUFF(ICAR,2) = PUF(ICAR,1)
           ENDIF
           IF (RPUFF(ICAR,1) .EQ. 1.0 .AND. RPUFF(ICAR,3) .EQ. 0.
     1          .AND. IPUF(ICAR) .EQ. 0) THEN
              RPUFF(ICAR,3) = GLBTM
           ENDIF
           RPUFF(ICAR,4) = MAX(RPUFF(ICAR,4),PUF(ICAR,2))
```

```
         RPUFF(ICAR,5) = MAX(RPUFF(ICAR,5),PUF(ICAR,3))
      ENDIF

      IF (IPLUM(ICAR) .GT. 0) THEN
         IF (RPLUM(ICAR,1) .EQ. 0.) THEN
            RPLUM(ICAR,1) = 1.0
            RPLUM(ICAR,2) = PLM(ICAR,1)
         ENDIF

         IF (RPLUM(ICAR,1) .EQ. 1.0 .AND. RPLUM(ICAR,3) .EQ. 0.
     1         .AND. IPLUM(ICAR) .EQ. 0) THEN
            RPLUM(ICAR,3) = GLBTM
         ENDIF
         DO 100 K = 1,300
            RPLUM(ICAR,4) = MAX(RPLUM(ICAR,4),PLMM(K,1))
            RPLUM(ICAR,5) = MAX(RPLUM(ICAR,5),PLMM(K,2))
            IF (PLMM(K,1) .EQ. 0.) GOTO 101
100      CONTINUE
101      CONTINUE
      ENDIF
10000 END

************************************************************************
************************************************************************
```

```
************************************************************************
************************************************************************

      SUBROUTINE PRT()
      INTEGER            PATTR(14)
      CHARACTER*20       NAM
      INTEGER          NBLUE,GREEN,LBLUE,RED,VIOLET,BROWN,
     1                 WHITE,GRAY,BBLUE,BGREEN,BBBLUE,BRED,
     1                 BVIOLT,YELLOW,BWHITE
      COMMON/COLORS/   NBLUE,GREEN,LBLUE,RED,VIOLET,BROWN,
     1                 WHITE,GRAY,BBLUE,BGREEN,BBBLUE,BRED,
     1                 BVIOLT,YELLOW,BWHITE
      COMMON/BLK/      IBLACK

      CALL DRWIC2()
      PATTR(1) = 0
      PATTR(2) = 0
      PATTR(3) = 0
      PATTR(4) = 1
      PATTR(5) = 1
      PATTR(6) = 0
      PATTR(7) = 1
      PATTR(8) = 0
      PATTR(9) = 0
      IBLACK     =  15
      NBLUE     =  15
      GREEN     =  0
      LBLUE     =  15
      RED       =  15
      VIOLET    =  15
      BROWN     =  15
      WHITE     =  15
      GRAY      =  15
      BBLUE     =  15
      BGREEN    =  15
      BBBLUE    =  15
      BRED      =  15
      BVIOLT    =  15
      YELLOW    =  15
      BWHITE    =  15
      CALL DRWIC2()
      CALL SUBRDR()
      CALL SETVIE(0.002,0.002,.746875,.85,15,-1)
      CALL SETPRN('/HALOEPSN.PRN/')
      CALL GPRINT()
      DO 222 I = 1,10
         DO 222 J = 1,90000
222   CONTINUE
      CALL SETC()
      CALL DRWIC2()
      CALL SUBRDR()
10000 END
```

```
**************************************************************************
**************************************************************************

      SUBROUTINE REP()
      INTEGER          EVNT

      COMMON/DERDAT/   RCARS(40,13),NCARS,UWIND,UANGL,PATM,
     1                 TATM,UTRAIN,RCURV,ICARDR,ITRUCK,IAXLE,
     1                 CENTER,D1X,D1Y,D2X,D2Y,
     1                 USTR,UEND,DX,DY,IDTRK,ISPEED,CENX,CENY,
     1                 NUMTNK,DERAIL,DERTYP,ICAR,ATMOSP
      COMMON/WORLDC/   XB,YB,XT,YT,ILEVEL
      COMMON/RESULT1/  IBLAST(8),BLAST(8,7),IFLARE(8),FLAR(8,3),
     1                 IPUFF(8),PUF(8,3),ISP(8),SP(8,6),
     1                 IPLUM(8),PLM(8,2),PLMM(300,2),
     1                 IFIRE(8),FIRE(8,4),RUPDAT(8,5),IPLM(8),IPUF(8)
      COMMON/GLTM/     GLBTM
      COMMON/DET/      IDETAL(7)
      COMMON/OUTFL/    OUTFLG
      COMMON/TANK/     ITCAR(40)
      COMMON/MREC/     RSPL(8,8),RBLAST(8,9),RRUPT(8,6),RFIRE(8,6),
     1                 RFLAR(8,5),RPLUM(8,5),RPUFF(8,5)


C
C
C*****  PRINT OUT SPILL DATA
C
C
C


      WRITE(1,'(A)')' '
      ICAR = 0
      EVNT = 1
      DO 100 I = 1,NCARS
         IF (RCARS(I,1) .EQ. 2.) THEN
            ICAR = ICAR + 1
            IF (RSPL(ICAR,1) .EQ. 1.0) THEN
     WRITE(1,'(I4,X,F5.1,4X,A,3X,F6.1,3X,F6.2,3X\)')
     1     EVNT,RSPL(ICAR,3),'  -  ',RSPL(ICAR,4),RSPL(ICAR,8)
               EVNT = EVNT + 1
               DO 99 K = 1, (ICAR - 1)
                  WRITE(1,'(A\)')'  -  '
99             CONTINUE
               WRITE(1,'(A\)')' SP  '
               DO 98 K = ICAR , (NUMTNK-1)
                  WRITE(1,'(A\)')'  -  '
98             CONTINUE
               WRITE(1,'(X)')
            ENDIF
         ENDIF
100   CONTINUE


C
C
C*****   PRINT OUT RUPTURE DATA
C
C

      WRITE(1,'(X)')
      ICAR = 0
      DO 200 I = 1,NCARS
         IF (RCARS(I,1) .EQ. 2.) THEN
            ICAR = ICAR + 1
            IF (RRUPT(ICAR,1) .GE. 1.0) THEN
    WRITE(1,'(I4,X,F5.1,4X,A,3X,F6.1,3X\)')
     1     EVNT,RRUPT(ICAR,3),'  -  ',RRUPT(ICAR,1)
               IF(RCARS(I,10) .EQ. 2.) THEN
                  WRITE(1,'(F6.1,3X\)')RCARS(I,11)
               ELSE
                  WRITE(1,'(A,3X\)')'   -  '
               ENDIF
               EVNT = EVNT + 1
               DO 199 K = 1,(ICAR - 1)
                  WRITE(1,'(A\)')'  -  '
199            CONTINUE
               WRITE(1,'(A\)')' RP  '
               DO 198 K = ICAR , (NUMTNK-1)
                  WRITE(1,'(A\)')'  -  '
198            CONTINUE
               WRITE(1,'(X)')
            ENDIF
         ENDIF
200   CONTINUE


C
C
C*****   PRINT OUT FIRE DATA
C
C
C

      WRITE(1,'(X)')
      ICAR = 0
      DO 300 I = 1,NCARS
         IF (RCARS(I,1) .EQ. 2.) THEN
            ICAR = ICAR + 1
            IF (RFIRE(ICAR,1) .EQ. 1.0) THEN
    WRITE(1,'(I4,X,F5.1,4X,F5.1,3X,F6.1,3X,A,3X\)')
```

```
     1       EVNT,RFIRE(ICAR,3),RFIRE(ICAR,4),RFIRE(ICAR,2),
     1       '   -  '
                 EVNT = EVNT + 1
                 DO 299 K = 1,(ICAR - 1)
                    WRITE(1,'(A\)')'  -  '
299              CONTINUE
                 WRITE(1,'(A\)')' FR  '
                 DO 298 K = ICAR , (NUMTNK-1)
                    WRITE(1,'(A\)')'  -  '
298              CONTINUE
                 WRITE(1,'(X)')
              ENDIF
           ENDIF
300     CONTINUE


C
C
C**** PRINT OUT FLARE DATA
C
C
C
        WRITE(1,'(X)')
        ICAR = 0
        DO 400 I = 1,NCARS
           IF (RCARS(I,1) .EQ. 2.) THEN
              ICAR = ICAR + 1
              IF (RFLAR(ICAR,1) .EQ. 1.0) THEN
      WRITE(1,'(I4,X,F5.1,4X,F5.1,3X,F6.1,3X,A,3X\)')
     1       EVNT,RFLAR(ICAR,3),RFLAR(ICAR,4),RFLAR(ICAR,5),
     1       '   -  '
                 EVNT = EVNT + 1
                 DO 399 K = 1,(ICAR - 1)
                    WRITE(1,'(A\)')'  -  '
399              CONTINUE
                 WRITE(1,'(A\)')' FL  '
                 DO 398 K = ICAR , (NUMTNK-1)
                    WRITE(1,'(A\)')'  -  '
398              CONTINUE
                 WRITE(1,'(X)')
              ENDIF
           ENDIF
400     CONTINUE


C
C
C****   PRINT OUT BLAST DATA
C
C
C
        WRITE(1,'(X)')
        ICAR = 0
        DO 500 I = 1,NCARS
           IF (RCARS(I,1) .EQ. 2.) THEN
              ICAR = ICAR + 1
              IF (RBLAST(ICAR,1) .EQ. 1.0) THEN
      WRITE(1,'(I4,X,F5.1,4X,A,3X,F6.1,3X,F6.1,3X\)')
     1      EVNT,RBLAST(ICAR,3),'  -  ',RBLAST(ICAR,6),
     1      RBLAST(ICAR,9)
                 EVNT = EVNT + 1
                 DO 499 K = 1,(ICAR - 1)
                    WRITE(1,'(A\)')'  -  '
499              CONTINUE
                 WRITE(1,'(A\)')' BL  '
                 DO 498 K = ICAR , (NUMTNK-1)
                    WRITE(1,'(A\)')'  -  '
498              CONTINUE
                 WRITE(1,'(X)')
              ENDIF
           ENDIF
500     CONTINUE


C
C
C****   PRINT OUT PUFF DATA
C
C
C
        WRITE(1,'(X)')
        ICAR = 0
        DO 600 I = 1,NCARS
           IF (RCARS(I,1) .EQ. 2.) THEN
              ICAR = ICAR + 1
              IF (RPUFF(ICAR,1) .EQ. 1.0) THEN
      WRITE(1,'(I4,X,F5.1,4X,F5.1,3X,F6.1,3X,F6.1,3X\)')
     1     EVNT,RPUFF(ICAR,2),RPUFF(ICAR,3),RPUFF(ICAR,4),
     1     RPUFF(ICAR,5)
                 EVNT = EVNT + 1
                 DO 599 K = 1,(ICAR - 1)
                    WRITE(1,'(A\)')'  -  '
599              CONTINUE
                 WRITE(1,'(A\)')' PU  '
                 DO 598 K = ICAR , (NUMTNK-1)
                    WRITE(1,'(A\)')'  -  '
598              CONTINUE
                 WRITE(1,'(X)')
              ENDIF
           ENDIF
600     CONTINUE
```

```
C
C
C**** PRINT OUT PLUME DATA
C
C
C

      WRITE(1,'(X)')
      ICAR = 0
      DO 700 I = 1,NCARS
         IF (RCARS(I,1) .EQ. 2.) THEN
            ICAR = ICAR + 1
            IF (RPLUM(ICAR,1) .EQ. 1.0) THEN
     WRITE(1,'(I4,X,F5.1,4X,F5.1,3X,F6.1,3X,F6.1,3X\)')
    1      EVNT,RPLUM(ICAR,2),RPLUM(ICAR,3),RPLUM(ICAR,4),
    1      RPLUM(ICAR,5)
               EVNT = EVNT + 1
               DO 699 K = 1,(ICAR - 1)
                  WRITE(1,'(A\)')'  -  '
699            CONTINUE
               WRITE(1,'(A\)')' PL  '
               DO 698 K = ICAR , (NUMTNK-1)
                  WRITE(1,'(A\)')'  -  '
698            CONTINUE
               WRITE(1,'(X)')
            ENDIF
         ENDIF
700   CONTINUE
10000 END


************************************************************************
************************************************************************


      SUBROUTINE PUTPUT()
      REAL              TYPE
      CHARACTER*20      NAME


      INTEGER           NBLUE,GREEN,LBLUE,RED,VIOLET,BROWN,
    1                   WHITE,GRAY,BBLUE,BGREEN,BBBLUE,BRED,
    1                   BVIOLT,YELLOW,BWHITE
      LOGICAL           OUTFLG
      COMMON/COLORS/    NBLUE,GREEN,LBLUE,RED,VIOLET,BROWN,
    1                   WHITE,GRAY,BBLUE,BGREEN,BBBLUE,BRED,
    1                   BVIOLT,YELLOW,BWHITE
      COMMON/ARRAS/     INTARR(100,10),REARR(100,100,2),KOS
      COMMON/WORLDC/    XB,YB,XT,YT,ILEVEL
      COMMON/DERDAT/    RCARS(40,13),NCARS,UWIND,UANGL,PATM,
1                  TATM,UTRAIN,RCURV,ICARDR,ITRUCK,IAXLE,
1                  CENTER,D1X,D1Y,D2X,D2Y,
1                  USTR,UEND,DX,DY,IDTRK,ISPEED,CENX,CENY,
1                  NUMTNK,DERAIL,DERTYP,ICAR,ATMOSP
   COMMON/RESULT1/ IBLAST(8),BLAST(8,7),IFLARE(8),FLAR(8,3),
1                  IPUFF(8),PUF(8,3),ISP(8),SP(8,6),
1                  IPLUM(8),PLM(8,2),PLMM(300,2),
1                  IFIRE(8),FIRE(8,4),RUPDAT(8,5),IPLM(8),IPUF(8)
   COMMON/GLBTM/   IGLBTM
   COMMON/DET/     IDETAL(7)
   COMMON/MODE/    IMODE
   COMMON/OUTFL/   OUTFLG
   COMMON/WINDS/   WSPEED
   COMMON/TANK/    ITCAR(40)
   COMMON/REPNAM/  NAME

   WRITE(1,'(A\)')'1________________________________________',
   WRITE(1,'(A)')'_________________________________'
   WRITE(1,'(A,A)')' Title  -  ',NAME
   WRITE(1,'(A,A)')'+          ',NAME
   WRITE(1,'(A,A)')'+          ',NAME
   WRITE(1,'(A\)')' _________________________________________',
   WRITE(1,'(A)')'_________________________________'
   WRITE(1,'(A\)')' DERACS: Derailment Accident Simulator -',
   WRITE(1,'(A)')' Summary    '
   WRITE(1,'(A\)')' _________________________________________',
   WRITE(1,'(A)')'_________________________________'
   WRITE(1,'(A)')' '
   WRITE(1,'(A)')' '
   WRITE(1,'(A,F5.0,A,F5.0,A,F10.0,A,F5.0)')
1  ' Wind Speed: ',WSPEED,'   Wind Direction: ',UWIND,
1  '   Barometric Pressure: ',PATM
   WRITE(1,'(A,F5.0)')' Air Temperature: ',TATM,
   WRITE(1,'(A,I5,A,A,I5,A,I5)')' Train Speed:',ISPEED,
1    ' Km/Hr  ',' Derailment Leader: ',ICARDR,
1    '   Truck Number: ',ITRUCK
   WRITE(1,'(A\)')' _________________________________________',
   WRITE(1,'(A)')'_________________________________'
   WRITE(1,'(A)')' '
   WRITE(1,'(A)')' Train Consist Data:'
   WRITE(1,'(A)')' '
   WRITE(1,'(A\)')' Consist # |  Veh. type  |  Comm. | ',
   WRITE(1,'(A)')'Insul. | Init. Temp. |  Init. Fill',
   WRITE(1,'(A)')' '
   DO 100 I = 1, NCARS
      WRITE(1,'(I5,6X,I5,6X\)')I,INT(RCARS(I,1))
      IF (RCARS(I,1) .EQ. 2.) THEN
         WRITE(1,'(3X,I5,3X,I5,8X,F5.0,9X,F5.0)')
1    INT(RCARS(I,2)),INT(RCARS(I,3)),RCARS(I,5),RCARS(I,4)
      ELSE
         WRITE(1,'(3X,A,3X,A,8X,A,9X,A)')'    -','    -',
1                 '   - ',' - '
```

```
           ENDIF
100     CONTINUE


        WRITE(1,'(A\)')'1____________________________________________',
        WRITE(1,'(A)')'____________________________________'
        WRITE(1,'(A,A)')' Title  -  ',NAME
        WRITE(1,'(A,A)')'+          ',NAME
        WRITE(1,'(A,A)')'+          ',NAME
        WRITE(1,'(A\)')' ____________________________________________',
        WRITE(1,'(A)')'____________________________________'
        WRITE(1,'(A)')' DERACS: Derailment Accident Simulator -',
        WRITE(1,'(A,F6.1)')' Summary  -  Global Time: ',
     1            REAL(REAL(IGLBTM)/3.0)
        WRITE(1,'(A\)')' ____________________________________________',
        WRITE(1,'(A)')'____________________________________'
        WRITE(1,'(A)')' Events Summary:'
        WRITE(1,'(A)')' '
        WRITE(1,'(A\)')' Evt#  Start  End     Dim1      Dim2     '
        IR = 0
        DO 200 I = 1 , NCARS
           IF (RCARS(I,1) .EQ. 2.) THEN
              IR = IR + 1
              WRITE(1,'(A,I1,2X\)')'Tc',IR
           ENDIF
200     CONTINUE
        CALL REP()
        WRITE(1,'(A)')'1'
10000   END

************************************************************************
************************************************************************
```

```
*******************************************************************************
*******************************************************************************

      SUBROUTINE PPOST()
$STORAGE:2
      INTEGER         SWITCH,ICNUM,PATTR(9)
      REAL            XR,YR
      LOGICAL         OUTFLG,CLICK,OUTF
      CHARACTER*20    PRINTER,NAME

      COMMON/ARRAS/   INTARR(100,10),REARR(100,100,2),KOS
      COMMON/WORLDC/  XB,YB,XT,YT,ILEVEL
      COMMON/DERDAT/  RCARS(40,13),NCARS,UWIND,UANGL,PATM,
     1                TATM,UTRAIN,RCURV,ICARDR,ITRUCK,IAXLE,
     1                CENTER,D1X,D1Y,D2X,D2Y,
     1                USTR,UEND,DX,DY,IDTRK,ISPEED,CENX,CENY,
     1                NUMTNK,DERAIL,DERTYP,ICAR,ATMOSP
      COMMON/GLBTM/   IGLBTM
      COMMON/DET/     IDETAL(7)
      COMMON/MODE/    IMODE
      COMMON/OUTFL/   OUTFLG
      COMMON/OUT/     OUTF
      COMMON/WINDS/   WSPEED
      COMMON/TRKK/    ITRACK
      COMMON/TANK/    ITCAR(40)
      COMMON/BLK/     IBLACK
      COMMON/REPNAM/  NAME
      COMMON/POSIT/   XP,YP

      DERAIL = 2.
      IMODE = 2
      IGLBTM = 0
      WRITE(*,*)' 2J5;5H1;4;33mDERACS0m'
      WRITE(*,*)' 10;5H1m ENTER NAME OF REPORT FILE0m'
22    WRITE(*,'(A\)')' 12;10H --> '
      READ(*,'(A)',END=22,ERR=22)NAME
      WRITE(*,*)' 0;0H'
      CALL RCRS(0)
C     WRITE(*,*)' 16;10HPROCESSING DATA FILES'
      CALL PP()
      OPEN(5,FILE='DERACS.OUT',STATUS='OLD')

      CALL SETDEV('/HALOIBME.DEV/')
      CALL INITGR(4)
      CALL SETIEE(1)
      CALL SETASP(.75)
      CALL SETLOC(2,2)
      CALL SETLAT(0)
      CALL SETSCR(1)
      CALL DISPLA(2)
      CALL SETVIE(0.,0.,.746875,.85,4,2)
      OUTFLG = .FALSE.
      OUTF = .FALSE.
      CALL RESETW()
      XP = 50.
      YP = 175.

      CALL SUBRDR()
      CALL DISPLA(1)
      CALL DRWIC2()
      CALL UPDAT2()

      CALL RESETW()
40    CALL READLO(XR,YR,SWITCH)
      CALL MOVHCA(XR,YR)

      IF(XR .LT. XT) THEN
         CONTINUE
      ELSE
50       CALL WHICH2(ICNUM,CLICK)
         IF (CLICK) THEN
            IF (ICNUM .EQ. 1 ) THEN
               CALL SUBZOM(2)
               CALL DRWIC2()
            ENDIF
            IF (ICNUM .EQ. 2 ) THEN
               CALL SUBZOM(1)
               CALL DRWIC2()
            ENDIF
            IF (ICNUM .EQ. 3 ) CALL DETAIL()
            IF (ICNUM .EQ. 4 ) CALL PRT()
            IF (ICNUM .EQ. 5 ) CALL KEY()
            IF (ICNUM .EQ. 6 ) CALL START()
            IF (ICNUM .EQ. 9 ) THEN
               OUTF = .TRUE.
               GOTO 900
            ENDIF
            CALL UPDAT2()
         ENDIF
         IF (OUTF) GOTO 900
         CALL READLO(XR,YR,SWITCH)
         IF (SWITCH .EQ. 2 .OR. SWITCH .EQ. 130) THEN
             CALL WAIT2()
             CALL RESETW()
             GOTO 40
         ENDIF
         GOTO 50
      ENDIF
      GOTO 40

900   CONTINUE
      CALL CLOSEG()
      OPEN(1,FILE=NAME,STATUS='NEW')
      CALL PUTPUT()
      WRITE(*,*)' 2J 2;5H End of DERACS Simulation'
      WRITE(*,*)' 2J 5;5H Press return to print Report'
      WRITE(*,*)' 9;5H   <Q> To QUIT'
11    WRITE(*,'(A\)')' 10;5H '
      READ(*,'(A)',END=12,ERR=12)JUNK
      IF (JUNK .EQ. 'Q' .OR. JUNK .EQ. 'q') GOTO 10000
12    WRITE(*,*)' 2J 23;0H'
      OPEN(1,FILE='LPT1:',STATUS='NEW')
      CALL PUTPUT()
      OPEN(1,FILE='CON',STATUS='NEW')
      CALL PUTPUT()

10000 CALL CLOSEG()
      WRITE(*,*)' 2J10;10H1mDERACS   0mSESSION    COMPLETE'
      END

*************************************************************************
*************************************************************************


      SUBROUTINE KEY()
      INTEGER     SWITCH
      CALL SETSCR(2)
      CALL GREAD('/KEY.IMG/')
      CALL SETSCR(1)
      CALL DISPLA(2)
10    CALL READLO(X,Y,SWITCH)
      IF (SWITCH .EQ. 130 .OR. SWITCH .EQ. 2 .OR. SWITCH .EQ.
     1    132 .OR. SWITCH .EQ. 4) GOTO 900
      GOTO 10
900   CALL DISPLA(1)
10000 END




*************************************************************************
*************************************************************************



      SUBROUTINE DETAIL()
      COMMON/DET/      IDETAL(7)


      CALL SETVIE(.75,.2,.99,.7,0,1)
      CALL SETWOR(0.,30.,15.,0.)
      CALL SETTEX(1,1,0,0)
      CALL SETTCL(12,2)
      CALL MOVTCA(1.,2.)
      CALL TEXT('/SHOW DETAIL/')
      CALL MOVTCA(1.,2.)
      CALL TEXT('/___________/')
5     IF (IDETAL(1) .EQ. 0) THEN
         CALL SETTCL(8,2)
         CALL SETTCL(8,2)
      ELSE
         CALL SETTCL(12,2)
         CALL SETTCL(12,2)
      ENDIF
      CALL MOVTCA(1.,6.)
      CALL TEXT('/1.BLAST/')
      IF (IDETAL(2) .EQ. 0) THEN
         CALL SETTCL(8,2)
         CALL SETTCL(8,2)
      ELSE
         CALL SETTCL(12,2)
         CALL SETTCL(12,2)
      ENDIF
      CALL MOVTCA(1.,8.)
      CALL TEXT('/2.BLAST THERMAL/')
      IF (IDETAL(3) .EQ. 0) THEN
         CALL SETTCL(8,2)
         CALL SETTCL(8,2)
      ELSE
         CALL SETTCL(12,2)
         CALL SETTCL(12,2)
      ENDIF
      CALL MOVTCA(1.,10.)
      CALL TEXT('/3.FIRE THERMAL/')
      IF (IDETAL(4) .EQ. 0) THEN
         CALL SETTCL(8,2)
         CALL SETTCL(8,2)
      ELSE
         CALL SETTCL(12,2)
         CALL SETTCL(12,2)
      ENDIF
      CALL MOVTCA(1.,12.)
      CALL TEXT('/4.PUFF/')
      IF (IDETAL(5) .EQ. 0) THEN
         CALL SETTCL(8,2)
         CALL SETTCL(8,2)
      ELSE
         CALL SETTCL(12,2)
         CALL SETTCL(12,2)
      ENDIF
      CALL MOVTCA(1.,14.)
      CALL TEXT('/5.PLUME/')
      IF (IDETAL(6) .EQ. 0) THEN
         CALL SETTCL(8,2)
         CALL SETTCL(8,2)
      ELSE
```

```
            CALL SETTCL(12,2)
            CALL SETTCL(12,2)
         ENDIF
         CALL MOVTCA(1.,16.)
         CALL TEXT('/6.FLARES/')
         IF (IDETAL(7) .EQ. 0) THEN
            CALL SETTCL(8,2)
            CALL SETTCL(8,2)
         ELSE
            CALL SETTCL(12,2)
            CALL SETTCL(12,2)
         ENDIF
         CALL MOVTCA(1.,18.)
         CALL TEXT('/7.SPILLS/')

         CALL SETTCL(12,2)
         CALL SETTCL(12,2)
         CALL MOVTCA(1.,23.)
         CALL TEXT('/8.EXIT/')
         CALL MOVTCA(1.,26.)
         CALL TEXT('/0.RESET/')
         CALL DELTCU()
55       CALL PROMPT()
         READ(*,'(I10)',ERR=55,END=55)IDET
         CALL CPROM()
         IF (IDET .EQ. 8) GOTO 10000
         IF (IDET .EQ. 0) GOTO 900
         IF (ABS(IDET) .GT. 8) GOTO 55
         IF (IDET .LT. 0) THEN
            ITMP = ABS(IDET)
            IDETAL(ITMP) = 0
         ELSE
            IDETAL(IDET) = 1
         ENDIF
         GOTO 5
900      CONTINUE
         DO 111 KK = 1,7
            IDETAL(KK) = 1
111      CONTINUE
         GOTO 5

10000    CALL DRWIC2()
         CALL SUBRDR()
         END

**************************************************************************
**************************************************************************


         SUBROUTINE  WHICH2(ICNUM,CLICK)
         INTEGER       ICNUM
```

```
      LOGICAL       CLICK
      INTEGER       SWITCH
      REAL          X,Y

      COMMON/WORLDC/   XB,YB,XT,YT,ILEVEL
      COMMON/POSIT/    XP,YP


      CLICK = .FALSE.
      CALL LEFT2()
      CALL ORGLOC(XP,YP)
      CALL MOVHCA(XP,YP)
40    CALL READLO(X,Y,SWITCH)
      CALL MOVHCA(X,Y)
      IF (X .LT. 0.) THEN
         X = 2.
      ELSEIF (Y .LT. 0.) THEN
         Y = 2.
      ELSEIF (X .GT. 100.) THEN
         X = 98.
      ELSEIF (Y .GT. 250.) THEN
         Y = 248.
      ENDIF
      IF (SWITCH .EQ. 4 .OR. SWITCH .EQ. 132) THEN
            XP = X
            YP = Y
            CALL FICON2(Y,ICNUM)
            IF (ICNUM .EQ. 0) GOTO 40
            CLICK = .TRUE.
            CALL WAIT4()
            GOTO 10000
      ELSEIF (SWITCH .EQ. 2 .OR. SWITCH .EQ. 130) THEN
            GOTO 10000
      ENDIF
      GOTO 40
10000 END

*************************************************************************
*************************************************************************

      SUBROUTINE FICON2(Y,ICNUM)
      REAL        Y
      INTEGER     ICNUM
      IF     (Y .GE. 60. .AND. Y .LT. 70.)   THEN
            ICNUM = 1
      ELSEIF (Y .GE. 80. .AND. Y .LT. 90.)   THEN
            ICNUM = 2
      ELSEIF (Y .GE. 100. .AND. Y .LT. 110.) THEN
            ICNUM = 3
      ELSEIF (Y .GE. 120. .AND. Y .LT. 130.) THEN
            ICNUM = 4
      ELSEIF (Y .GE. 140. .AND. Y .LT. 150.) THEN
```

```
               ICNUM = 5
         ELSEIF (Y .GE. 160. .AND. Y .LT. 170.) THEN
               ICNUM = 6
         ELSEIF (Y .GE. 180. .AND. Y .LT. 190.) THEN
               ICNUM = 7
         ELSEIF (Y .GE. 200. .AND. Y .LT. 210.) THEN
               ICNUM = 8
         ELSEIF (Y .GE. 220. .AND. Y .LT. 230.) THEN
               ICNUM = 9
         ELSE
               ICNUM = 0
         ENDIF
10000    END

************************************************************************
************************************************************************

         SUBROUTINE UPDAT2()

         COMMON/ARRAS/    INTARR(100,10),REARR(100,100,2),KOS
         COMMON/WORLDC/   XB,YB,XT,YT,ILEVEL
         COMMON/DERDAT/   RCARS(40,13),NCARS,UWIND,UANGL,PATM,
     1                    TATM,UTRAIN,RCURV,ICARDR,ITRUCK,IAXLE,
     1                    CENTER,D1X,D1Y,D2X,D2Y,
     1                    USTR,UEND,DX,DY,IDTRK,ISPEED,CENX,CENY,
     1                    NUMTNK,DERAIL,DERTYP,ICAR,ATMOSP
         COMMON/GLBTM/    IGLBTM
         COMMON/DET/      IDETAL(7)
         COMMON/WINDS/    WSPEED


         WRITE(*,'(A,F4.1,A)')
     1       ' 19;62HTIME: ',REAL(REAL(IGLBTM)/3.0),' MIN.'
         WRITE(*,'(A,F4.1)')' 20;62HWIND SPEED:',WSPEED
         WRITE(*,'(A,F4.1)')' 21;62HWIND DIREC:',UWIND
         CALL SETLEV()
         WRITE(*,'(A,I1,A,I2,A,I2)')' 23;0HZOOM: ',ILEVEL,
     1             '  TANK CARS: ',NUMTNK,'  TOTAL CARS: ',NCARS
         WRITE(*,'(A,F6.0,A,F6.0,A,F6.0,A,F6.0,A)')' 24;0HWORLD: ',
     1        XB,',',YB,',',XT,',',YT,
     1        ' 22;0H'

10000    END

************************************************************************
************************************************************************

         SUBROUTINE DRWIC2()

         INTEGER          NBLUE,GREEN,LBLUE,RED,VIOLET,BROWN,
     1                    WHITE,GRAY,BBLUE,BGREEN,BBBLUE,BRED,
     1                    BVIOLT,YELLOW,BWHITE
```

```
      COMMON/COLORS/   NBLUE,GREEN,LBLUE,RED,VIOLET,BROWN,
     1                 WHITE,GRAY,BBLUE,BGREEN,BBBLUE,BRED,
     1                 BVIOLT,YELLOW,BWHITE
      COMMON/BLK/      IBLACK

      CALL CLRMEN()
      CALL SETCOL(GREEN)
      CALL SETLNW(1)
      CALL SETLNS(1)
      CALL SETHAT(1)
      CALL SETXOR(0)
      CALL SETVIE(.746875,0.,1.,.7,0,GREEN)
      CALL SETWOR(0.,250.,100.,0.)

************   TITLE - DERAC
      CALL SETCOL(IBLACK)
      CALL SETCOL(IBLACK)
      CALL SETTEX(2,2,0,0)
      CALL SETTCL(RED,GRAY)
      CALL MOVTCA(20.,20.)
      CALL TEXT('/DERACS/')

***   OPTIONS
      CALL SETTEX(1,1,0,0)
      CALL SETTCL(BROWN,GRAY)
      CALL MOVTCA(30.,40.)
      CALL TEXT('/OPTIONS/')
      CALL MOVTCA(30.,40.)
      CALL TEXT('/_______/')

      CALL SETTCL(IBLACK,GRAY)
      CALL MOVTCA(30.,70.)
      CALL TEXT('/ZOOM IN/')

      CALL MOVTCA(30.,90.)
      CALL TEXT('/ZOOM OUT/')

      CALL MOVTCA(30.,110.)
      CALL TEXT('/DETAIL/')

      CALL MOVTCA(30.,130.)
      CALL TEXT('/SNAPSHOT/')

      CALL MOVTCA(30.,150.)
      CALL TEXT('/KEY/')

      CALL MOVTCA(30.,170.)
      CALL TEXT('/START/')

      CALL MOVTCA(30.,190.)
      CALL TEXT('//')
      CALL MOVTCA(30.,210.)
      CALL TEXT('//')

      CALL MOVTCA(30.,230.)
      CALL TEXT('/EXIT/')
      CALL UPDAT2()
      CALL LEFT2()
10000 END


************************************************************************
************************************************************************


      SUBROUTINE CLRMEN()

      CALL SETLNW(1)
      CALL SETLNS(1)
      CALL SETHAT(1)
      CALL SETXOR(0)
      CALL SETVIE(.746875,.7,1.,1.,0,0)
10000 END


************************************************************************
************************************************************************

      SUBROUTINE LEFT2()
      INTEGER         NBLUE,GREEN,LBLUE,RED,VIOLET,BROWN,
     1                WHITE,GRAY,BBLUE,BGREEN,BBBLUE,BRED,
     1                BVIOLT,YELLOW,BWHITE
      COMMON/COLORS/  NBLUE,GREEN,LBLUE,RED,VIOLET,BROWN,
     1                WHITE,GRAY,BBLUE,BGREEN,BBBLUE,BRED,
     1                BVIOLT,YELLOW,BWHITE

      CALL SETVIE(.746875,0.,1.,.7,-1,-1)
      CALL SETWOR(0.,250.,100.,0.)
      CALL INITHC(23.,9.,0)
10000 END

************************************************************************
************************************************************************

      SUBROUTINE START()
      INTEGER        SWITCH
      LOGICAL        OUTFLG
      REAL           X,Y
      COMMON/GLBTM/    IGLBTM
      COMMON/OUTFL/    OUTFLG
```

```
10       CALL READLO(X,Y,SWITCH)
         IF (SWITCH .EQ. 4 .OR. SWITCH .EQ. 132 .OR. SWITCH .EQ.
     1       130 .OR. SWITCH .EQ. 2) THEN
               CALL WAIT4()
               CALL WAIT2()
               GOTO 10000
         ELSE
               IF (OUTFLG) GOTO 9
               CALL SETVIE(.78,.4,.98,.55,0,1)
               CALL SETWOR(0.,10.,10.,0.)
               CALL SETTEX(1,1,0,0)
               CALL SETTCL(12,2)
               CALL MOVTCA(2.5,5.)
               CALL TEXT('/DRAWING/')
               CALL DELTCU()
               CALL NEXTST()
9              IF (OUTFLG) THEN
                  CALL DRWIC2()
                  CALL DWIN()
                  CALL SETTEX(1,1,0,0)
                  CALL SETTCL(12,2)
                  CALL MOVTCA(1.,2.)
                  CALL TEXT('/END OF DERACS/')
                  CALL MOVTCA(1.,4.)
                  CALL TEXT('/SIMULATION!/')
                  CALL MOVTCA(1.,8.)
                  CALL TEXT('/HIT RETURN TO/')
                  CALL MOVTCA(1.,10.)
                  CALL TEXT('/CONTINUE./')
                  CALL DELTCU()
11                CALL PROMPT()
                  READ(*,'(I2)',END=12,ERR=12)JUNK
12                CALL CPROM()
                  CALL DRWIC2()
                  CALL UPDAT2()
                  GOTO 10000
               ENDIF
               CALL DRWIC2()
               CALL UPDAT2()
               IGLBTM = IGLBTM + 1
        ENDIF
        GOTO 10
10000   END

************************************************************************
************************************************************************

        SUBROUTINE PP()
        REAL JUNK,TIME

        OPEN(21,FILE='TCAR1.PUF',STATUS='OLD')
```

```
      READ(21,*,END=20)JUNK
      READ(21,*,END=20)TIME
      IF (JUNK .NE. -1.0) THEN
         CLOSE(21)
         OPEN(21,FILE='PUFF1.ALT',STATUS='OLD',ACCESS='DIRECT',
     1        RECL=8)
      ENDIF

20    CONTINUE
      OPEN(22,FILE='TCAR2.PUF',STATUS='OLD')
      READ(22,*,END=30)JUNK
      READ(22,*,END=30)TIME
      IF (JUNK .NE. -1.0) THEN
         CLOSE(22)
         OPEN(22,FILE='PUFF2.ALT',STATUS='OLD',ACCESS='DIRECT',
     1        RECL=8)
      ENDIF



30    CONTINUE
      OPEN(23,FILE='TCAR3.PUF',STATUS='OLD')
      READ(23,*,END=40)JUNK
      READ(23,*,END=40)TIME
      IF (JUNK .NE. -1.0) THEN
         CLOSE(23)
         OPEN(23,FILE='PUFF3.ALT',STATUS='OLD',ACCESS='DIRECT',
     1        RECL=8)
      ENDIF



40    CONTINUE
      OPEN(24,FILE='TCAR4.PUF',STATUS='OLD')
      READ(24,*,END=50)JUNK
      READ(24,*,END=50)TIME
      IF (JUNK .NE. -1.0) THEN
         CLOSE(24)
         OPEN(24,FILE='PUFF4.ALT',STATUS='OLD',ACCESS='DIRECT',
     1        RECL=8)
      ENDIF



50    CONTINUE
      OPEN(25,FILE='TCAR5.PUF',STATUS='OLD')
      READ(25,*,END=60)JUNK
      READ(25,*,END=60)TIME
      IF (JUNK .NE. -1.0) THEN
         CLOSE(25)
         OPEN(25,FILE='PUFF5.ALT',STATUS='OLD',ACCESS='DIRECT',
     1        RECL=8)
```

```
      ENDIF


60    CONTINUE
      OPEN(26,FILE='TCAR6.PUF',STATUS='OLD')
      READ(26,*,END=70)JUNK
      READ(26,*,END=70)TIME
      IF (JUNK .NE. -1.0) THEN
         CLOSE(26)
         OPEN(26,FILE='PUFF6.ALT',STATUS='OLD',ACCESS='DIRECT',
     1         RECL=8)
      ENDIF



70    CONTINUE
      OPEN(27,FILE='TCAR7.PUF',STATUS='OLD')
      READ(27,*,END=80)JUNK
      READ(27,*,END=80)TIME
      IF (JUNK .NE. -1.0) THEN
         CLOSE(27)
         OPEN(27,FILE='PUFF7.ALT',STATUS='OLD',ACCESS='DIRECT',
     1         RECL=8)
      ENDIF



80    CONTINUE
      OPEN(28,FILE='TCAR8.PUF',STATUS='OLD')
      READ(28,*,END=10000)JUNK
      READ(28,*,END=10000)TIME
      IF (JUNK .NE. -1.0) THEN
         CLOSE(28)
         OPEN(28,FILE='PUFF8.ALT',STATUS='OLD',ACCESS='DIRECT',
     1         RECL=8)
      ENDIF

10000 END
```

```
      SUBROUTINE DRAW1()
$STORAGE:2
      INTEGER           TYPE,SP
      CHARACTER         INSUL,JUNK
      INTEGER           NBLUE,GREEN,LBLUE,RED,VIOLET,BROWN,
     1                  WHITE,GRAY,BBLUE,BGREEN,BBBLUE,BRED,
     1                  BVIOLT,YELLOW,BWHITE
      COMMON/COLORS/    NBLUE,GREEN,LBLUE,RED,VIOLET,BROWN,
     1                  WHITE,GRAY,BBLUE,BGREEN,BBBLUE,BRED,
     1                  BVIOLT,YELLOW,BWHITE
      COMMON/ARRAS/     INTARR(100,10),REARR(100,100,2),KOS
      COMMON/DERDAT/    RCARS(40,13),NCARS,UWIND,UANGL,PATM,
     1                  TATM,UTRAIN,RCURV,ICARDR,ITRUCK,IAXLE,
     1                  CENTER,D1X,D1Y,D2X,D2Y,
     1                  USTR,UEND,DX,DY,IDTNK,ISPEED,CENX,CENY,
     1                  NUMTNK,DERAIL,DERTYP,ICAR,ATMOSP

      IF (NCARS .GT. 0) THEN
         CALL DWIN()
         CALL SETTEX(1,1,0,0)
         CALL SETTCL(12,8)
         CALL MOVTCA(1.,2.)
         CALL TEXT('/TRAIN HAS /')
         CALL MOVTCA(1.,4.)
         CALL TEXT('/BEEN ENTERED!/')
         CALL MOVTCA(1.,8.)
         CALL TEXT('/<C> TO CONTINUE/')
         CALL MOVTCA(1.,10.)
         CALL TEXT('/<Q> TO QUIT/')
         CALL DELTCU()
99       CALL PROMPT()
         READ(*,'(A)',END=99,ERR=99)JUNK
         CALL CPROM()
         IF (JUNK .NE. 'C' .AND. JUNK .NE. 'c') THEN
            CALL DRICON()
            GOTO 10000
         ELSE
            NCARS = 0
            NUMTNK = 0
         ENDIF
      ENDIF

5     NCARS = NCARS + 1
      ICAR = NCARS
      CALL SETLNW(1)
      CALL SETCOL(0)
      CALL SETLNS(1)
      IF (NCARS .GT. 40) THEN
         CALL MAXCAR(1)
         GOTO 900
      ENDIF
      CALL DWIN()
      CALL SETTEX(1,1,0,0)
      CALL SETTCL(12,8)
      CALL MOVTCA(1.,2.)
      CALL TEXT('/ENTER TYPE /')
      CALL MOVTCA(1.,4.)
      CALL TEXT('/OF CAR;/')
      CALL MOVTCA(1.,8.)
      CALL TEXT('/1, 2 OR 3/')
      CALL MOVTCA(1.,10.)
      CALL TEXT('/9 TO EXIT/')
      CALL DELTCU()
10    CALL PROMPT()
      READ(*,'(I10)',END=10,ERR=10)ITMP
      RCARS(ICAR,1) = REAL(ITMP)
      IF ((RCARS(ICAR,1) .GT. 3. .OR. RCARS(ICAR,1) .LT. 1.)
     1      .AND. RCARS(ICAR,1) .NE. 9.)
     1       GOTO 10
      CALL CPROM()
      CALL DWIN()
      IF (RCARS(ICAR,1) .EQ. 9.) THEN
         CALL DRICON()
         GOTO 900
      ENDIF
      IF (RCARS(ICAR,1) .EQ. 2.) THEN
          NUMTNK = NUMTNK + 1
          IF (NUMTNK .GT. 40) THEN
             NUMTNK = 20
             CALL MAXCAR(2)
             GOTO 5
          ENDIF
          CALL SETTEX(1,1,0,0)
          CALL MOVTCA(1.,2.)
          CALL TEXT('/ENTER/')
          CALL MOVTCA(1.,4.)
          CALL TEXT('/COMMODITY;/')
          CALL MOVTCA(1.,6.)
          CALL TEXT('/1, 2 OR 3/')
          CALL DELTCU()
20        CALL PROMPT()
          READ(*,'(I10)',END=20,ERR=20)ITMP
          RCARS(ICAR,2) = REAL(ITMP)
          IF (RCARS(ICAR,2) .GT. 3. .OR. RCARS(ICAR,2) .LT. 1.)
     1         GOTO 20
          CALL CPROM()

          CALL DWIN()
          CALL MOVTCA(1.,2.)
          CALL TEXT('/IS CAR INSULATED ?/')
          CALL MOVTCA(1.,4.)
          CALL TEXT('/1.YES  0.NO/')
          CALL DELTCU()
30        WRITE(*,'(A\)')' 23;60H
```

```
          WRITE(*,'(A\)')' 23;63H'
          WRITE(*,'(A\)')' -> 0'
          WRITE(*,'(A\)')' 23;63H'
          WRITE(*,'(A\)')' -> '
          READ(*,'(I10)',END=30,ERR=30)ITMP
          RCARS(ICAR,3) = REAL(ITMP)
          IF (RCARS(ICAR,3) .GT. 1. .OR. RCARS(ICAR,3) .LT. 0.)
     1             GOTO 30
          CALL CPROM()
          CALL DWIN()
          CALL MOVTCA(1.,2.)
          CALL TEXT('/ENTER INITIAL/')
          CALL MOVTCA(1.,4.)
          CALL TEXT('/FILL LEVEL/')
          CALL MOVTCA(1.,6.)
          CALL TEXT('/....(%)..../')
          CALL DELTCU()
50        WRITE(*,'(A\)')' 23;60H                              '
          WRITE(*,'(A\)')' 23;63H'
          WRITE(*,'(A\)')' -> 0'
          WRITE(*,'(A\)')' 23;63H'
          WRITE(*,'(A\)')' -> '
          READ(*,'(F12.8)',ERR=50,END=50)RCARS(ICAR,4)
          IF (RCARS(ICAR,4) .LT. .01)
     1       RCARS(ICAR,4)  = RCARS(ICAR,4) * 10**8
          IF (RCARS(ICAR,4) .GT. 100. .OR. RCARS(ICAR,4) .LT. 0.)
     1            GOTO 50
          CALL CPROM()
          CALL DWIN()
          CALL MOVTCA(1.,2.)
          CALL TEXT('/ENTER INITIAL/')
          CALL MOVTCA(1.,4.)
          CALL TEXT('/TEMPERATURE/')
          CALL MOVTCA(1.,6.)
          CALL TEXT('/..degrees C../')
          CALL DELTCU()
60        WRITE(*,'(A\)')' 23;60H                              '
          WRITE(*,'(A\)')' 23;63H'
          WRITE(*,'(A\)')' -> 0'
          WRITE(*,'(A\)')' 23;63H'
          WRITE(*,'(A\)')' -> '
          READ(*,'(F12.8)',ERR=60,END=60)RCARS(ICAR,5)
          IF (RCARS(ICAR,5) .LT. .01)
     1       RCARS(ICAR,5)  = RCARS(ICAR,5) * 10**8
          IF (RCARS(ICAR,5) .GT. 40. .OR. RCARS(ICAR,5) .LT. -40.)
     1            GOTO 60
          CALL CPROM()
          CALL DWIN()
      ENDIF
      CALL UPDATE()
      CALL DRC()
      GOTO 5
```

```
900     IF (NCARS .LE. 1) THEN
           NCARS = 0
        ELSE
           NCARS = NCARS - 1
        ENDIF
        IF (NCARS .GE. 1) THEN
           CALL FIXLN()
           CALL SETVIE(.76,.4,.99,.75,0,1)
           CALL SETWOR(0.,14.,10.,0.)
           CALL MOVTCA(1.,2.)
           CALL TEXT('/CONFIRM TRAIN/')
           CALL MOVTCA(1.,4.)
           CALL TEXT('/DESCRIPTION!/')
           CALL MOVTCA(1.,10.)
           CALL TEXT('/<C> TO CONTINUE/')
           CALL MOVTCA(1.,12.)
           CALL TEXT('/<R> TO REDO/')
           CALL DELTCU()
4          CALL PROMPT()
           READ(*,'(A)',END=4,ERR=4)JUNK
           CONTINUE
           CALL CPROM()
           CALL DRICON()
           CALL SUBRDR()
           IF (JUNK .EQ. 'C' .OR. JUNK .EQ. 'c') THEN
              CONTINUE
           ELSEIF (JUNK .EQ. 'R' .OR. JUNK .EQ. 'r') THEN
              NCARS = 0
              NUMTNK = 0
              GOTO 5
           ELSE
              GOTO 4
           ENDIF
           CALL SUBTRN()
        ENDIF
10000   END


**********************************************************************
**********************************************************************


        SUBROUTINE DRC()
        INTEGER            NBLUE,GREEN,LBLUE,RED,VIOLET,BROWN,
     1                     WHITE,GRAY,BBLUE,BGREEN,BBBLUE,BRED,
     1                     BVIOLT,YELLOW,BWHITE
        COMMON/COLORS/     NBLUE,GREEN,LBLUE,RED,VIOLET,BROWN,
     1                     WHITE,GRAY,BBLUE,BGREEN,BBBLUE,BRED,
     1                     BVIOLT,YELLOW,BWHITE
        COMMON/ARRAS/      INTARR(100,10),REARR(100,100,2),KOS
        COMMON/DERDAT/     RCARS(40,13),NCARS,UWIND,UANGL,PATM,
     1                     TATM,UTRAIN,RCURV,ICARDR,ITRUCK,IAXLE,
     1                    CENTER,D1X,D1Y,D2X,D2Y,
     1                    USTR,UEND,DX,DY,IDTRK,ISPEED,CENX,CENY,
     1                    NUMTNK,DERAIL,DERTYP,ICAR,ATMOSP

      CALL SETXOR(0)
      CALL SETVIE(0.,0.75,.746875,.85,GREEN,GREEN)
      CALL SETWOR(0.,10.,10000.,0.)
      DO 100 I = 1, ICAR
         CALL DRCARS(I)
100   CONTINUE
10000 END


************************************************************************
************************************************************************


      SUBROUTINE DRCARS(I)
      INTEGER            I

      INTEGER            NBLUE,GREEN,LBLUE,RED,VIOLET,BROWN,
     1                   WHITE,GRAY,BBLUE,BGREEN,BBBLUE,BRED,
     1                   BVIOLT,YELLOW,BWHITE
      COMMON/COLORS/     NBLUE,GREEN,LBLUE,RED,VIOLET,BROWN,
     1                   WHITE,GRAY,BBLUE,BGREEN,BBBLUE,BRED,
     1                   BVIOLT,YELLOW,BWHITE
      COMMON/DERDAT/    RCARS(40,13),NCARS,UWIND,UANGL,PATM,
     1                  TATM,UTRAIN,RCURV,ICARDR,ITRUCK,IAXLE,
     1                  CENTER,D1X,D1Y,D2X,D2Y,
     1                  USTR,UEND,DX,DY,IDTRK,ISPEED,CENX,CENY,
     1                  NUMTNK,DERAIL,DERTYP,ICAR,ATMOSP

      IF (RCARS(I,1) .EQ. 1. .OR. RCARS(I,1) .EQ. 3.) THEN
         CALL SETHAT(1)
         CALL SETHAT(1)
         CALL SETCOL(GRAY)
         CALL SETCOL(GRAY)
         CALL BOX(I*200.,1.,I*200.+150.,6.)
         GOTO 10000
      ELSEIF (RCARS(I,1) .EQ. 2.) THEN
         CALL SETCOL(YELLOW)
         CALL SETCOL(YELLOW)
         CALL SETCOL(YELLOW)
         CALL SETCOL(YELLOW)
         IF (RCARS(I,2) .EQ. 1.) THEN
            CALL SETHAT(4)
         ELSEIF (RCARS(I,2) .EQ. 2.) THEN
            CALL SETHAT(5)
         ELSEIF (RCARS(I,2) .EQ. 3.) THEN
            CALL SETHAT(3)
         ENDIF
      ENDIF
      CALL BAR(I*200.,1.,I*200.+150.,6.)
      CALL BOX(I*200.,1.,I*200.+150.,6.)
10000 END

*****************************************************************************
*****************************************************************************

      SUBROUTINE SUBTRN()

      INTEGER          SPEED
      REAL             WX1,WY1,WX2,WY2
      INTEGER          NCARS
      INTEGER          NBLUE,GREEN,LBLUE,RED,VIOLET,BROWN,
     1                 WHITE,GRAY,BBLUE,BGREEN,BBBLUE,BRED,
     1                 BVIOLT,YELLOW,BWHITE
      COMMON/ARRAS/    INTARR(100,10),REARR(100,100,2),KOS
      COMMON/COLORS/   NBLUE,GREEN,LBLUE,RED,VIOLET,BROWN,
     1                 WHITE,GRAY,BBLUE,BGREEN,BBBLUE,BRED,
     1                 BVIOLT,YELLOW,BWHITE
      COMMON/DERDAT/   RCARS(40,13),NCARS,UWIND,UANGL,PATM,
     1                 TATM,UTRAIN,RCURV,ICARDR,ITRUCK,IAXLE,
     1                 CENTER,D1X,D1Y,D2X,D2Y,
     1                 USTR,UEND,DX,DY,IDTRK,ISPEED,CENX,CENY,
     1                 NUMTNK,DERAIL,DERTYP,ICAR,ATMOSP

      CALL SETLNW(1)
      CALL SETCOL(0)
      CALL SETLNS(1)
      CALL DWIN()
      CALL SETTEX(1,1,0,0)
      CALL SETTCL(12,8)
      CALL MOVTCA(1.,2.)
      CALL TEXT('/TRAIN DATA/')
      CALL MOVTCA(1.,2.)
      CALL TEXT('/__________/')
      CALL MOVTCA(1.,4.)
      CALL TEXT('/ENTER/')
      CALL MOVTCA(1.,6.)
      CALL TEXT('/TRAIN SPEED;/')
      CALL MOVTCA(1.,8.)
      CALL TEXT('|....Km/Hr....|')
      CALL DELTCU()
20    CALL PROMPT()
      READ(*,'(I10)',ERR=20,END=20)ISPEED
      IF (ISPEED .GT. 100 .OR. ISPEED .LT. 1) THEN
         GOTO 20
      ENDIF
      CALL CPROM()
      CALL DWIN()
      CALL MOVTCA(1.,2.)
      CALL TEXT('/TRAIN DATA/')
      CALL MOVTCA(1.,2.)
      CALL TEXT('/__________/')
      CALL MOVTCA(1.,4.)
      CALL TEXT('/DERAILMENT/')
      CALL MOVTCA(1.,6.)
      CALL TEXT('/LEADER/')
      CALL MOVTCA(1.,8.)
      CALL TEXT('/-CAR NUMBER-/')
      CALL DELTCU()
30    CALL PROMPT()
      READ(*,'(I10)',END=30,ERR=30)ICARDR
      IF (ICARDR .GT. NCARS .OR. ICARDR .LT. 1) GOTO 30
      CALL CPROM()
      CALL DWIN()
      CALL MOVTCA(1.,2.)
      CALL TEXT('/TRAIN DATA/')
      CALL MOVTCA(1.,2.)
      CALL TEXT('/__________/')
      CALL MOVTCA(1.,4.)
      CALL TEXT('/DERAILMENT/')
      CALL MOVTCA(1.,6.)
      CALL TEXT('/LEADER/')
      CALL MOVTCA(1.,8.)
      CALL TEXT('/-TRUCK NUMBER-/')
      CALL DELTCU()
40    CALL PROMPT()
      READ(*,'(I10)',END=40,ERR=40)ITRUCK
      IF (ITRUCK .NE. 1 .AND. ITRUCK .NE. 2) THEN
          GOTO 40
      ENDIF
      IF (ITRUCK .EQ. 1) THEN
         ITRUCK = 1
      ELSE
         ITRUCK = -(1)
      ENDIF
      CALL CPROM()
      CALL DWIN()
      CALL MOVTCA(1.,2.)
      CALL TEXT('/TRAIN DATA/')
      CALL MOVTCA(1.,2.)
      CALL TEXT('/__________/')
      CALL MOVTCA(1.,4.)
      CALL TEXT('/DERAILMENT/')
      CALL MOVTCA(1.,6.)
      CALL TEXT('/LEADER/')
      CALL MOVTCA(1.,8.)
      CALL TEXT('/-AXLE NUMBER-/')
      CALL DELTCU()
50    CALL PROMPT()
      READ(*,'(I10)',END=50,ERR=50)IAXLE
      IF (IAXLE .NE. 1 .AND. IAXLE .NE. 2) THEN
          GOTO 50
      ENDIF
```

```
       CALL CPROM()
       CALL DRICON()
       CALL DELTCU()
10000  END

***************************************************************************
***************************************************************************

       SUBROUTINE MAXCAR(PARAM)
       INTEGER              PARAM
       REAL                 X,Y
       INTEGER              SWITCH
       INTEGER              NBLUE,GREEN,LBLUE,RED,VIOLET,BROWN,
     1                      WHITE,GRAY,BBLUE,BGREEN,BBBLUE,BRED,
     1                      BVIOLT,YELLOW,BWHITE
       COMMON/COLORS/       NBLUE,GREEN,LBLUE,RED,VIOLET,BROWN,
     1                      WHITE,GRAY,BBLUE,BGREEN,BBBLUE,BRED,
     1                      BVIOLT,YELLOW,BWHITE

       CALL LEFT()
       IF (PARAM .EQ. 1) THEN
           CALL DWIN()
           CALL SETTEX(1,1,0,0)
           CALL SETTCL(12,8)
           CALL MOVTCA(1.,2.)
           CALL TEXT('/TRAIN EXCEEDS/')
           CALL MOVTCA(1.,4.)
           CALL TEXT('/MAXIMUM OF/')
           CALL MOVTCA(1.,6.)
           CALL TEXT('/40 CARS/')
           CALL DELTCU()
        ELSE
           CALL DWIN()
           CALL SETTEX(1,1,0,0)
           CALL SETTCL(12,8)
           CALL MOVTCA(1.,2.)
           CALL TEXT('/TANK CARS/')
           CALL MOVTCA(1.,4.)
           CALL TEXT('/EXCEED MAX./')
           CALL MOVTCA(1.,6.)
           CALL TEXT('/OF 8/')
           CALL DELTCU()
       ENDIF

10     CALL READLO(X,Y,SWITCH)
       IF (SWITCH .EQ. 4 .OR. SWITCH .EQ. 132) THEN
            CALL DRICON()
            CALL WAIT4()
            GOTO 10000
       ELSE
            GOTO 10
       ENDIF
```

```
10000   END
```

```
      SUBROUTINE SUBZOM(TYPE)
$STORAGE:2
      INTEGER          TYPE
      INTEGER          SWITCH,NLEVEL
      REAL             X,Y,DIST,XINC,YINC,XDEC,YDEC
      COMMON/WORLDC/   XB,YB,XT,YT,ILEVEL


      CALL SETLNW(1)
      CALL SETLNW(1)
      CALL SETCOL(0)
      CALL SETCOL(0)
      CALL SETLNS(1)
      CALL SETLNS(1)
      CALL SETLEV()
      IF (TYPE .EQ. 1) THEN
         IF (ILEVEL .EQ. 1) GOTO 10000
         IF (ILEVEL .EQ. 2) THEN
            XT = 10000.
            YT = 10000.
            XB = 0.
            YB = 0.
         ELSEIF (ILEVEL .EQ. 3) THEN
            XINC = 450.
            YINC = 450.
            XDEC = -(450.)
            YDEC = -(450.)
            IF (XB .LT. 450.) THEN
               XINC = XINC + (450. - XB)
            ENDIF
            IF (YB .LT. 450.) THEN
               YINC = YINC + (450. - YB)
            ENDIF
            IF (XT .GT. 9550.) THEN
               XDEC = XDEC - (XT - 9550.)
            ENDIF
            IF (YT .GT. 9550.) THEN
               YDEC = YDEC - (YT - 9550.)
            ENDIF
            XT = XT + XINC
            YT = YT + YINC
            XB = XB + XDEC
            YB = YB + YDEC
         ENDIF
      ELSEIF (TYPE .EQ. 2) THEN
         IF (ILEVEL .EQ. 3) GOTO 10000
         IF (ILEVEL .EQ. 1) THEN
           DIST = 500.
         ELSEIF (ILEVEL .EQ. 2) THEN
           DIST = 100.
         ENDIF
         CALL ZWNDOW()
```

```
            CALL RESETW()
            CALL SETXOR(1)
            CALL SETXOR(1)
            CALL SETCOL(10)
            CALL SETCOL(10)
            CALL SETLNS(1)
            CALL SETLNS(1)
            CALL SETLNW(3)
            CALL SETLNW(3)
10          CALL READLO(X,Y,SWITCH)
            CALL DISXY(X,Y)

            IF ((X+DIST) .GE. XT) THEN
                X = XT - DIST
            ELSEIF ((X-DIST) .LE. XB) THEN
                X = XB + DIST
            ELSEIF ((Y+DIST) .GE. YT) THEN
                Y = YT - DIST
            ELSEIF ((Y-DIST) .LE. YB) THEN
                Y = YB + DIST
            ENDIF
            CALL ORGLOC(X,Y)
            CALL RBOX(X-DIST,Y-DIST,X+DIST,Y+DIST)
            IF (SWITCH .EQ. 4 .OR. SWITCH .EQ. 132) THEN
                CALL CLRXY()
                CALL SETXOR(0)
                CALL DELBOX()
                XB = X - DIST
                XT = X + DIST
                YB = Y - DIST
                YT = Y + DIST
                GOTO 10000
            ENDIF
            GOTO 10
         ENDIF
10000    CALL RESETW()
         CALL SETLEV()
         CALL SUBRDR()
         END

****************************************************************************
****************************************************************************

         SUBROUTINE ZWNDOW()

         CALL DWIN()
         CALL SETTEX(1,1,0,0)
         CALL SETTCL(12,2)
         CALL MOVTCA(1.,2.)
         CALL TEXT('/POSITION/')
         CALL MOVTCA(1.,4.)
         CALL TEXT('/BOX TO AREA/')
```

```
       CALL MOVTCA(1.,6.)
       CALL TEXT('/OF ZOOM/')
       CALL DELTCU()
10000  END

****************************************************************************
****************************************************************************
```

APPENDIX C

USER GUIDE

1.0 INTRODUCTION

1.1 This software was developed to simulate railway accidents. The system is divided into 4 separate modules which include the Preprocessor, Derailment Simulation, DERACS, and the Postprocessor. The operation of each module is explained in detail later in this manual.

The modules are executed from a DOS batch file and hence can be run separately from the DOS prompt. However, each module is dependent upon data created by the previous module; therefore, running the modules separately will produce unpredictable results.

1.2 The following are descriptions of the terminology used throughout this manual.

right click left click	- press the right (left) button on the mouse
">"	- represents the DOS prompt
cursor	- the crosshair cursor or pointer used by the mouse. When the mouse moves, the cursor tracks with it.
menu area	- the area at the right side of the screen containing menu icons.
drawing board	- the large green area of the screen. The graphic output is displayed here.
status area	- the black strip across the bottom of the screen. Continually being updated to show the status of DERACS.
ready mode	- DERACS is waiting for a command from the main menu.

2.0 REQUIREMENTS

2.1 System Requirements

The system requirements for running DERACS are as follows:

IBM AT or compatible with 80287 math coprocessor

Monitor	-	High Resolution Colour (640 x 350 pixels)
	-	Extended Graphics Adapter
Hard Disk	-	20 MB or greater
Floppy Disk	-	1.2 MB
Printer	-	Epson or compatible

2.2 Installation Procedures:

Included on the DERACS disks is a file called "INSTALL.BAT". This file will load all files from the distribution disks onto your hard drive. The install program will create a directory called \DERACS on your hard drive. If this directory already exists, the DERACS file will be copied into the existing directory. Any files with the same name will be overwritten.

Note: At some point, the DERACS distribution diskettes should be copied for backup. The disks are NOT copy protected and can be duplicated using DOS commands.

For details on this, consult your DOS manual.

When the install program is finished, remove the distribution disks and store them in a safe place.

3.0 MOUSE OPERATION

3.1 Operating the Mouse:

The mouse is used extensively as a means of interacting with the software. The DERACS software talks to a two button mouse but can, and does, talk to many different locator devices.

When the DERACS program is in Ready mode, the mouse is attached to a cursor. Moving the mouse will make the cursor track the same path on the screen.

Note: The mouse driver must be loaded into memory before running DERACS. If you fail to do this and you activate DERACS, your machine will have to be REBOOTED "<CTRL><ALT><DEL>".

For details on loading your mouse or locator driver, please consult your manual.

4.0 PREPROCESSOR

4.1 Preprocessor Overview:

The preprocessor module is designed to allow you to create a derailment scene and set certain conditions, which affect the output of the simulation. The preprocessor is a user friendly interface between yourself and the simulation software. It lets you add and delete information on a derailment setup and save the data in a standard form.

The preprocessor's main screen has 3 areas. They are the drawing board, menus and status area. The drawing board is shown in center left of Figure C4.1. The menus and status area are shown at right side and bottom strip of Figure C4.1.

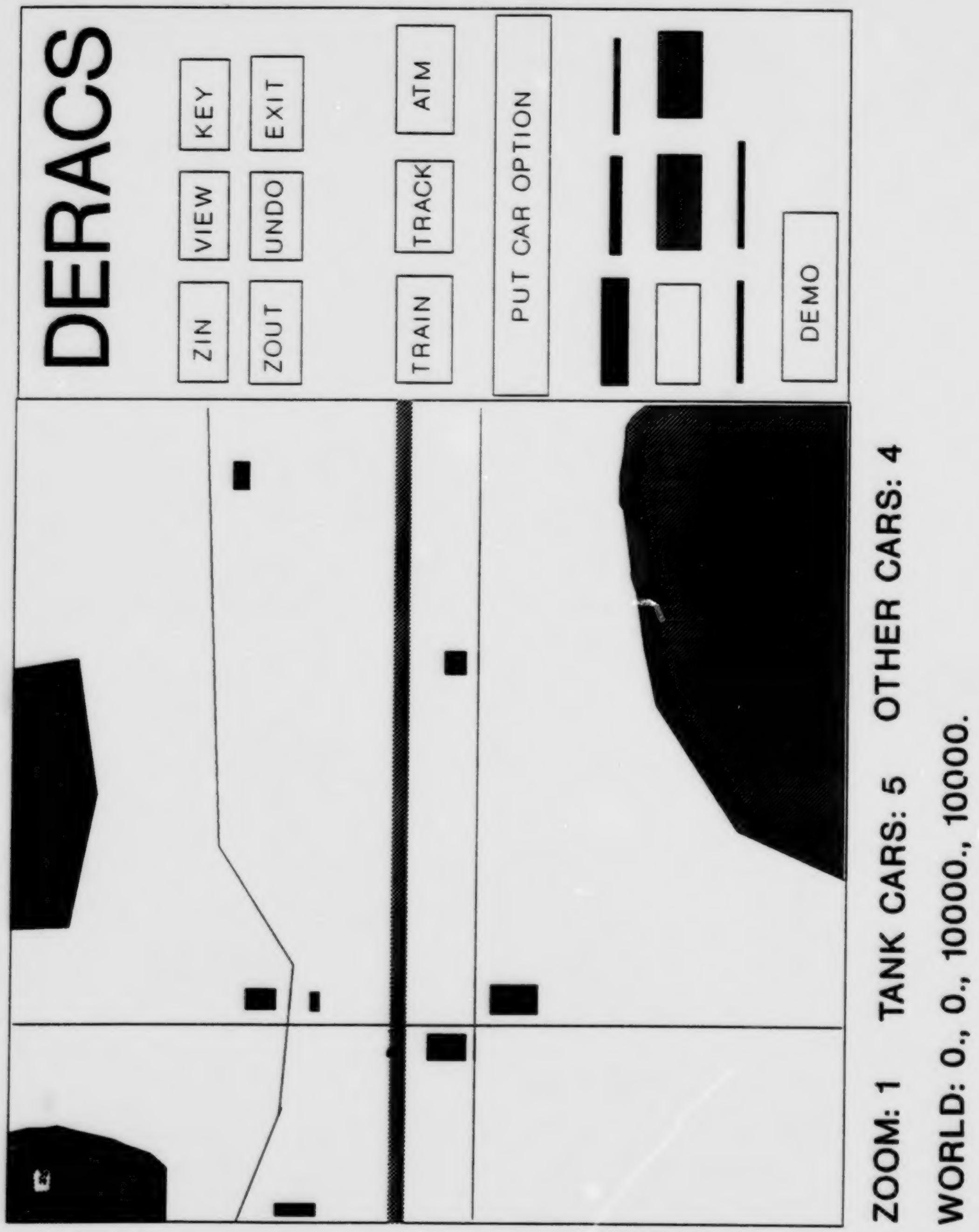
4.2 Operation:

Before starting DERACS make sure you are in the \DERACS directory. From the DOS prompt, type "RUN". This initiates a batch file which loads in the preprocessor. The main preprocessor screen will appear and you will have control of the cursor. At this point, you can choose any icon in the menu area. To do this, move the cursor to an icon and left click. After the command has been executed, the status area is updated to reflect any changes and you will be returned to ready mode. The cursor will remain in the menu area, but can be used as a pointer on the drawing board. To do this, right click and the cursor will jump to the middle of the drawing board. To get back to the menu area, simply move the cursor across the border.

4.3 Preprocessor Commands:

The following is a description of all the DERACS preprocessor commands.

COMMAND	DESCRIPTION
ZIN	This function will allow you to zoom on a particular spot on the screen. There are 3 zoom levels having the following dimensions: 1. 10,000 X 10,000 meters 2. 1,000 X 1,000 meters 3. 200 X 200 meters When this function is selected, a box will appear on the screen. Using the mouse, locate the box to surround the area of zoom. When the position is located, left click and the screen will be redrawn.


DERACS
ZIN
VIEW
KEY
ZOUT
UNDO
EXIT
TRAIN
TRACK
ATM
PUT CAR OPTION
DEMO

ZOOM: 1 TANK CARS: 5 OTHER CARS: 4

WORLD: 0., 0., 10000., 10000.

FIGURE C4.1 PREPROCESSOR DRAWING BOARD

ZOUT	This function allows you to reverse the zoom in process. When this function is selected, the drawing dimensions are altered and the screen is redrawn.
VIEW	VIEW - This command redraws the screen. This command is optional.
KEY	KEY PAGE - This command displays a legend of preprocessor symbols. When it is selected, the current screen clears and the legend appears. To return to the preprocessor main screen, left or right click. The key page is shown in Figure C4.2. This command is optional.
UNDO	This command decrements the drawing list counter. The entries in the list are deleted in a last on first off manner. You can undo until the drawing list equals zero. Each time an entry is deleted, the screen is redrawn to reflect the change. Note: You can only undo visual effect data.
EXIT	This command exits the preprocessor allowing the batch file to continue loading the next module. Before exiting, a check is made for missing data. Missing data consists of TRAIN, TRACK and ATM commands. If any of these are missing, you will be reminded and prompted for the necessary information. At this point, if you want to abort the preprocessor, enter "Q" at the prompt. Entering "Q" will abort the preprocessor, not saving any data.

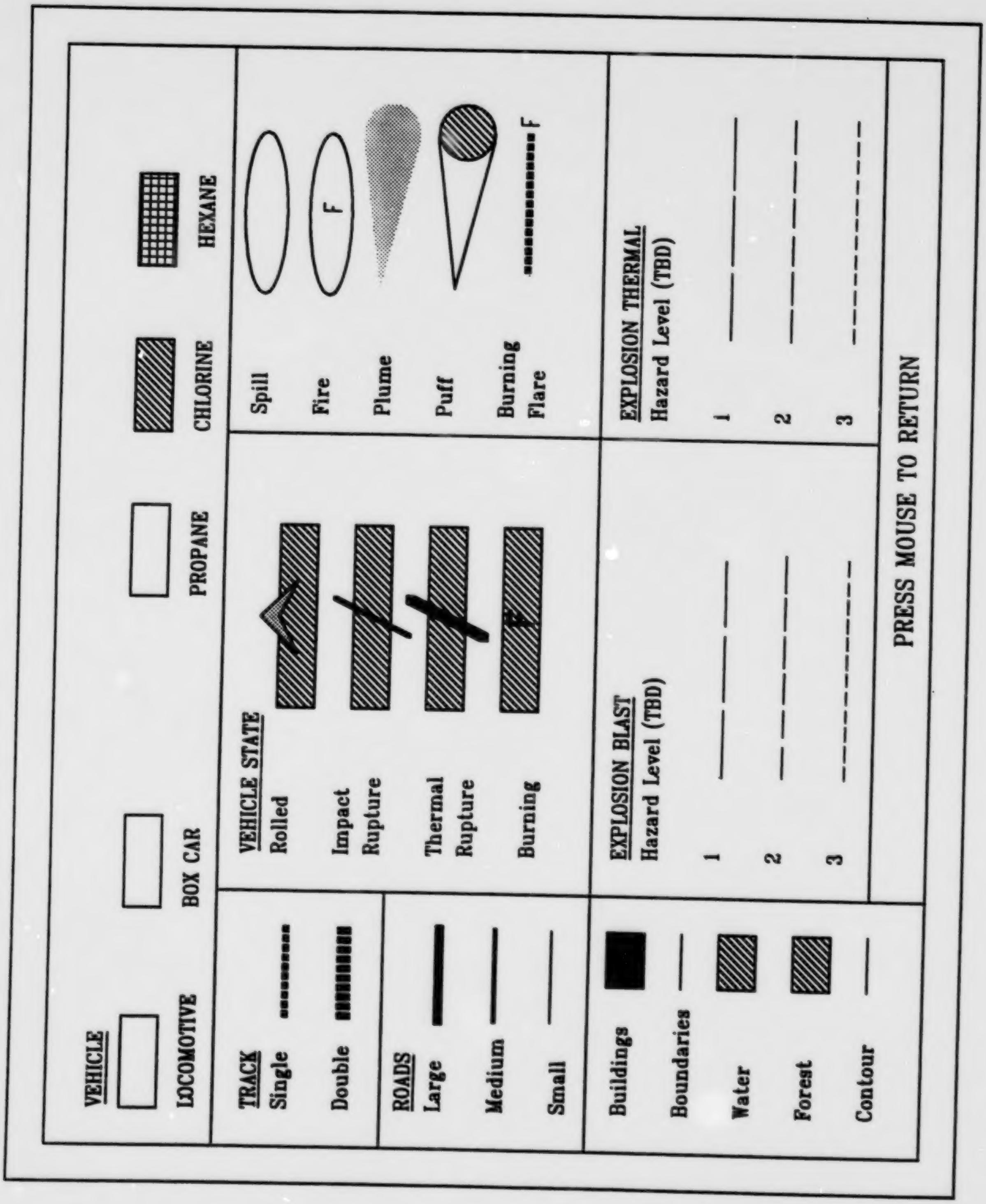

VEHICLE
LOCOMOTIVE
BOX CAR
PROPANE
CHLORINE
HEXANE
TRACK
Single
Double
ROADS
Large
Medium
Small
VEHICLE STATE
Rolled
Impact Rupture
Thermal Rupture
Burning
F
Spill
Fire
F
Plume
Puff
Burning Flare
F
Buildings
Boundaries
Water
Forest
Contour
EXPLOSION BLAST
Hazard Level (TBD)
1
2
3
EXPLOSION THERMAL
Hazard Level (TBD)
1
2
3
PRESS MOUSE TO RETURN

FIGURE C4.2 KEY PAGE

TRACK

There are 3 types of track you can input. The three types are DERAILMENT, JOINED and OTHER. They are described separately in the following paragraphs.

DERAILMENT TRACK

This function is used to enter the section of track where the derailment takes place. The derailment track is unique and therefore requires more description than other types of track. The derailment track describes the train direction and direction of force which will make the train derail on a particular side of the track.

The train will always travel from derailment track point #1 to derailment track point #2. Refer to Figure C4.3 for a visual representation of these conventions.

1. TANGENT TRACK - When this function is selected, you are asked to enter point #1. Using the mouse, you locate point #1 and left click. You then locate point #2 and left click. When point #2 is located and entered, the screen is redrawn and you are asked to locate the derailment point on the track. You do this by moving the mouse in the positive and negative x direction. As you do this, the cursor will follow the track. When a derailment point is located, left click. You will now be returned to ready mode.

2. CURVED TRACK - When this function is selected, you are asked to locate the center of curvature. When it is located, left click. You are then asked to manually key in the radius of curvature in meters. The next two inputs are START angle and END angle. The start and end angles can be any positive angle between 0 and 360 degrees. The convention for all angles entered is 0 degrees, is 3 o'clock and increases counterclockwise. When they are entered, the derailment track is drawn and you are asked to locate the derailment point. You can do this by moving the mouse in the positive x axis. When the derailment point is located, left click. You are now returned to ready mode.

Note: At any time you can re-enter the derailment track, overwriting the previous track.

JOINED

This command allows you to connect a section of visual track as described above. When this command is chosen, you are asked to key in the END number. You are then anchored to the derailment section and can complete the drawing of visual track.

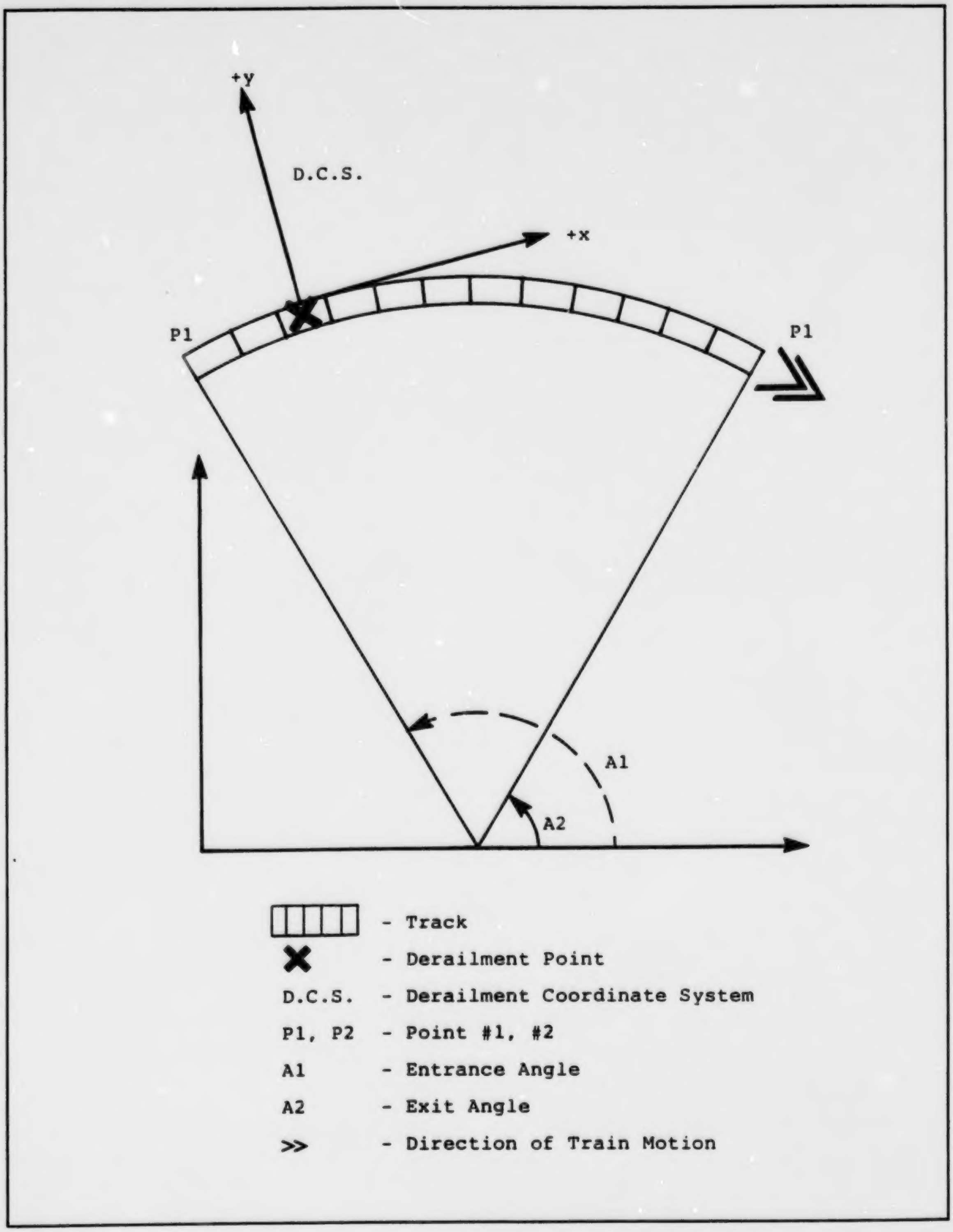

+y
D.C.S.
+x
P1
P1
A1
A2
- Track
- Derailment Point
D.C.S. - Derailment Coordinate System
P1, P2 - Point #1, #2
A1 - Entrance Angle
A2 - Exit Angle
>> - Direction of Train Motion

FIGURE C4.3 DRAWING OF DERAILMENT TRACK

OTHER	This command allows you to enter visual track. Visual track entry is described in Section 4.4 - Line Drawing. The Derailment Track command is mandatory.
TRAIN	TRAIN CONSIST DATA - This command allows you to describe the makeup of the train involved in the derailment and enter the train number, truck number and axle number of the car that causes the derailment. When this command is selected, a window appears in the menu area. The prompts are shown in Figure C.4.4. In prompt number 1 of Figure C4.4, the car types 1, 2, and 3 refer to locomotive, tank and box car respectively. In prompt number 2 the commodities refer to propane, chlorine and hexane. As the train description is entered, you will see a box appear along the bottom of the drawing board. Each box represents a car in the train. The maximum number of cars is 40. The placement of cars is performed by either the derailment program (Section 5.1) or derailment option in the preprocessor menu. This command is mandatory.
ATM	ATMOSPHERIC DATA - This command allows you to enter the atmospheric conditions affecting the derailment scene. The prompts appear in Figure C4.5. The units for wind speed are km/hr. The units for wind direction are in degrees from 3 o'clock counterclockwise. The temperature is in degrees C. The atmospheric pressure is in pascals where "normal" pressure is 101,000 pascals. This command is mandatory.

1.

```
ENTER TYPE
OF CAR
1,2 OR 3
9 TO EXIT
```

2.

```
ENTER
COMMODITY
1,2 OR 3
```

3.

```
IS CAR
INSULATED
0. NO  1. YES
```

4.

```
ENTER INITIAL
FILL LEVEL
.....(%).....
```

5.

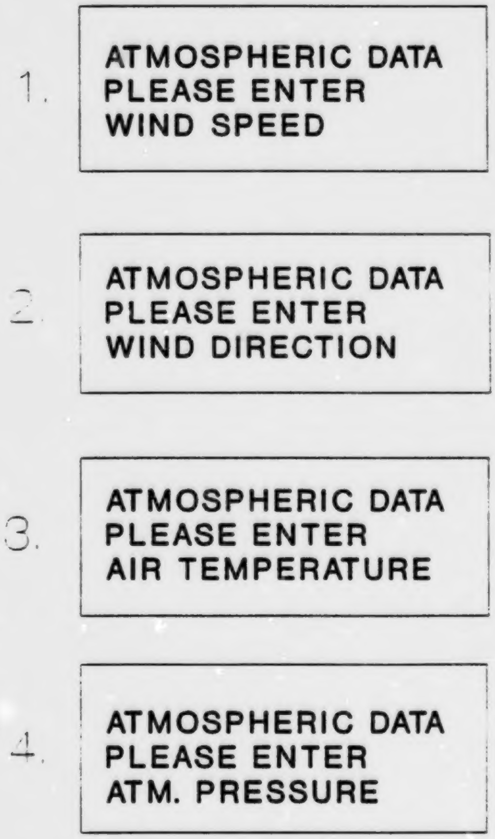
```
ENTER INITIAL
TEMPERATURE
...degrees C...
```

LOOP REPEATED UNTIL
TYPE EQUALS 9
TYPE:
1. LOCOMOTIVE
2. TANK
3. BOX

COMMODITY:
1. CHLORINE
2. PROPANE
3. HEXANE

NOTE:
BOXED ITEMS 2-5 ARE FOR TANK CARS ONLY

Figure C4.4 Train Consist Data



1\.
ATMOSPHERIC DATA
PLEASE ENTER
WIND SPEED

2\.
ATMOSPHERIC DATA
PLEASE ENTER
WIND DIRECTION

3\.
ATMOSPHERIC DATA
PLEASE ENTER
AIR TEMPERATURE

4\.
ATMOSPHERIC DATA
PLEASE ENTER
ATM. PRESSURE

FIGURE C4.5 ATMOSPHERIC CONDITIONS

PUT CAR OPTIONS

This command allows you to enter derailment data, therefore bypassing the derailment program. The input logic is illustrated as Figure C4.6. The car position is measured relative to the derailment coordinate system (D.C.S.). The car angle is measured in the D.C.S. having 0 degrees at 3 o'clock, increasing counterclockwise. The roll angle is positive in the direction of derailment. The rupture diameter is measured in meters and the distance is measured in meters from the centre of the car. A positive distance is forward of centre. The height is measured from the middle axis of the car. A positive height is above middle.

When all data has been entered, you can then view the placement of cars by zooming in on their location.

4.4 DRAWING:

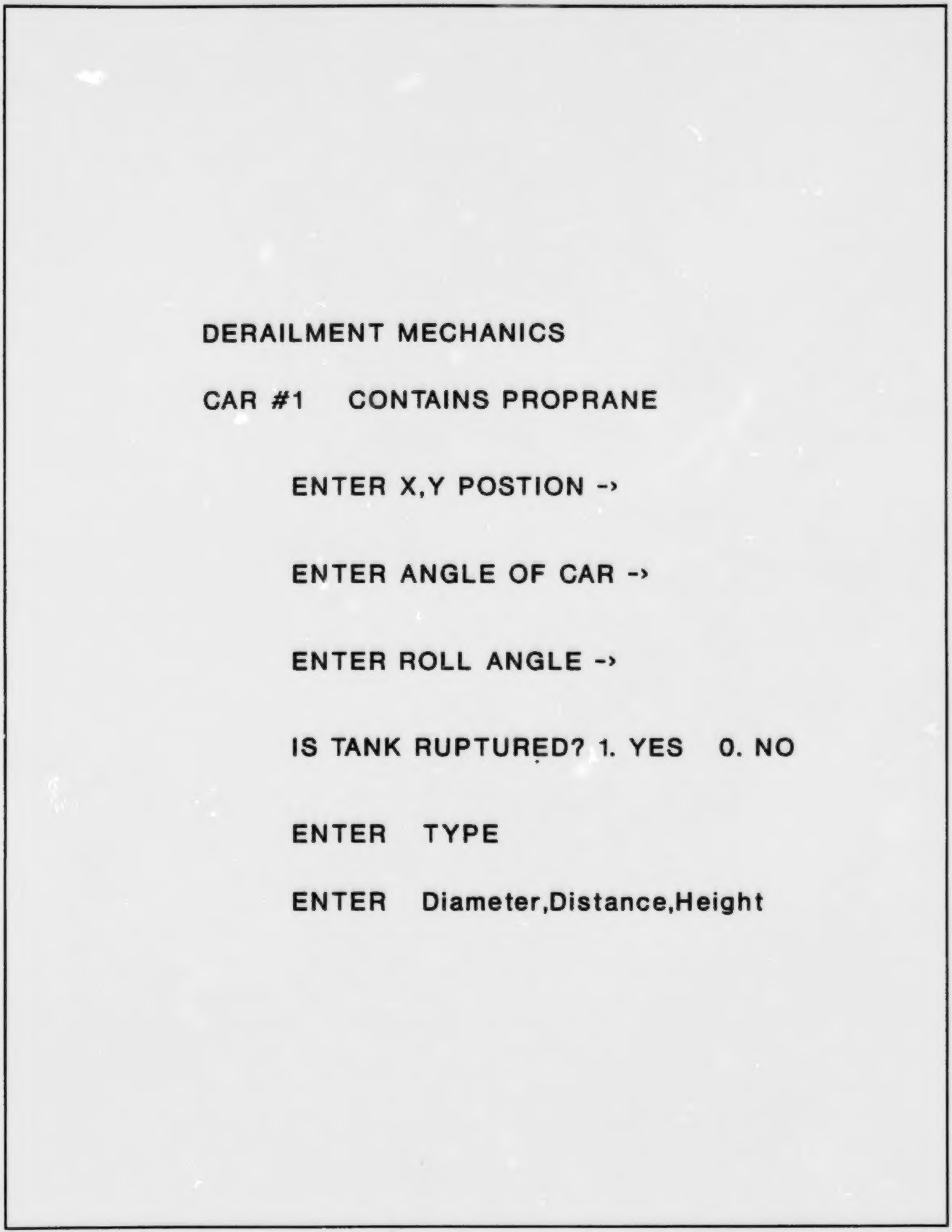
The methods of entering surrounding data is the same as drawing a picture. You are given the tools to construct a scene consisting of water, forests, roads, track, boundaries, buildings and contours. The procedure for drawing is essentially the same for all items. The following is a description of the procedures you need to construct a scene.

Line Drawing - Knowing how to draw series of connected lines is the most important procedure. Lines are used for contours, boundaries, roads, tracks, water and forest. For this example, a road is drawn.

Select road from the menu. Move the cursor to the first point of the line. Left click the mouse and move to the end point of the line. As you move the mouse, a rubber line will track with the mouse. Left click the mouse again and move to the next end point. Again, the rubber line will track with the mouse. Repeat this until the complete set of lines are drawn. Right click and the line will be displayed as Road and you will be returned to Ready mode.

Contour Lines - Contour lines are drawn using the line drawing procedure. When the contours are entered, you will be prompted for an elevation. The elevation is in meters.

Water & Forest - Water and Forest have the same drawing procedure. Start drawing an enclosed area using the method outlined in line drawing. When the enclosure is drawn, move the cursor inside the enclosure. You must now locate fill points for the water or forest. Left click in several different spots around the inner edge of the enclosure and then right click. The fill points you



```
DERAILMENT MECHANICS

CAR #1    CONTAINS PROPRANE

     ENTER X,Y POSTION ->

     ENTER ANGLE OF CAR ->

     ENTER ROLL ANGLE ->

     IS TANK RUPTURED? 1. YES    0. NO

     ENTER   TYPE

     ENTER   Diameter,Distance,Height
```

FIGURE C4.6   DERAILMENT MECHANICS

have entered are used to judge whether or not to fill when the screen is redrawn. If the fill point is outside the zoom area selected, the point will not be filled.

Note: If you are entering fill points from zoom level 2 or 3, keep point approximately 1 cm from inner edge.

Building - Buildings are represented as filled rectangles. To draw a building, select the building icon and move the cursor to the position of the building. Left click to anchor the cursor and move the mouse until the building is the desired size. Left click again and the building will become permanent.

4.5 DEMO:

The distribution disks contain a set of a demonstration files that are copied into a directory called \DERACS\DEMO on your hard disk. To access them, type "DEMO#" at the DOS prompt while in the \DERAC directory. The "#" sign represents the demonstration number. Your distribution contains DEMO1 through DEMO5.

When the files have been copied, type RUNDEMO from the DOS prompt. RUNDEMO will start the simulation program and then load the postprocessor.

Note: Changing the train, track or car placement in the preprocessor at this point will create unpredictable results.

4.6 SAVING & EXITING:

DERACS data files are automatically saved when you exit DERACS. The data file names are defined by the preprocessor and can only be changed using DOS commands. However, if you rename any files, the other modules will not be able to access them.

The exit command will check to see if all mandatory data has been entered. If data is missing, you will be reminded and instructed to enter the necessary information. Upon completion of said information, the DERACS data files will be created and control is returned to the batch file. The batch file will check if the file DERAIL.DAT was created. If it was not created, the Derailment program must be run. If DERAIL.DAT does exist, the Derailment program will be skipped.

4.7 SUMMARY:

The following is a step-by-step example of a preprocessor session:

1. Start preprocessor
2. Draw large roads at full zoom (level 1)
3. ZOOM IN on desired derailment area (level 2)

4. Draw DERAILMENT TRACK
5. ZOOM OUT to full zoom (level 1)
6. Complete visual track using OTHER and JOINED TRACK
7. ZOOM IN on derailment scene (level 3)
8. Draw buildings and small roads
9. Construct TRAIN
10. Enter ATMOSPHERIC DATA
11. Input car placement using PUT CAR OPTIONS
12. EXIT preprocessor

5.0 DERAILMENT SIMULATION

5.1 DERAILMENT SIMULATION PROGRAM:

This module simulates a derailment of up to 40 cars. It predicts car position, roll angle and rupture data. As explained earlier, this module will be run if the file DERAIL.DAT is not created by the preprocessor. For more details on this module, consult the main body of the report.

5.2 DERACS PROGRAM:

This module simulates the events that occur after a derailment has happened. This module is the most important part of the DERACS system and is run during every simulation. For more details on this module, consult the main body of the report.

6.0 POSTPROCESSOR

6.1 INTRODUCTION

This module is a user friendly interface to the data created by the DERACS program. It presents raw data in the form of figures. It also gives text reports of a particular instant in time of the simulation. It allows the user to examine a derailment site up close or get the global picture. It is designed the same way as the preprocessor and has the same screen layout. See Figure C6.1.

6.2 OPERATION:

The postprocessor is the last module to be loaded by the batch file. When it is loaded, it must reorganize and reformat the data files created by the DERACS module. This process will take up to 10 minutes. When the files are reorganized and loaded, the main postprocessor screen appears. The cursor is attached to the mouse and the postprocessor waits for a command.

6.3 MENU COMMANDS:

The following is a description of each command in the menu area.

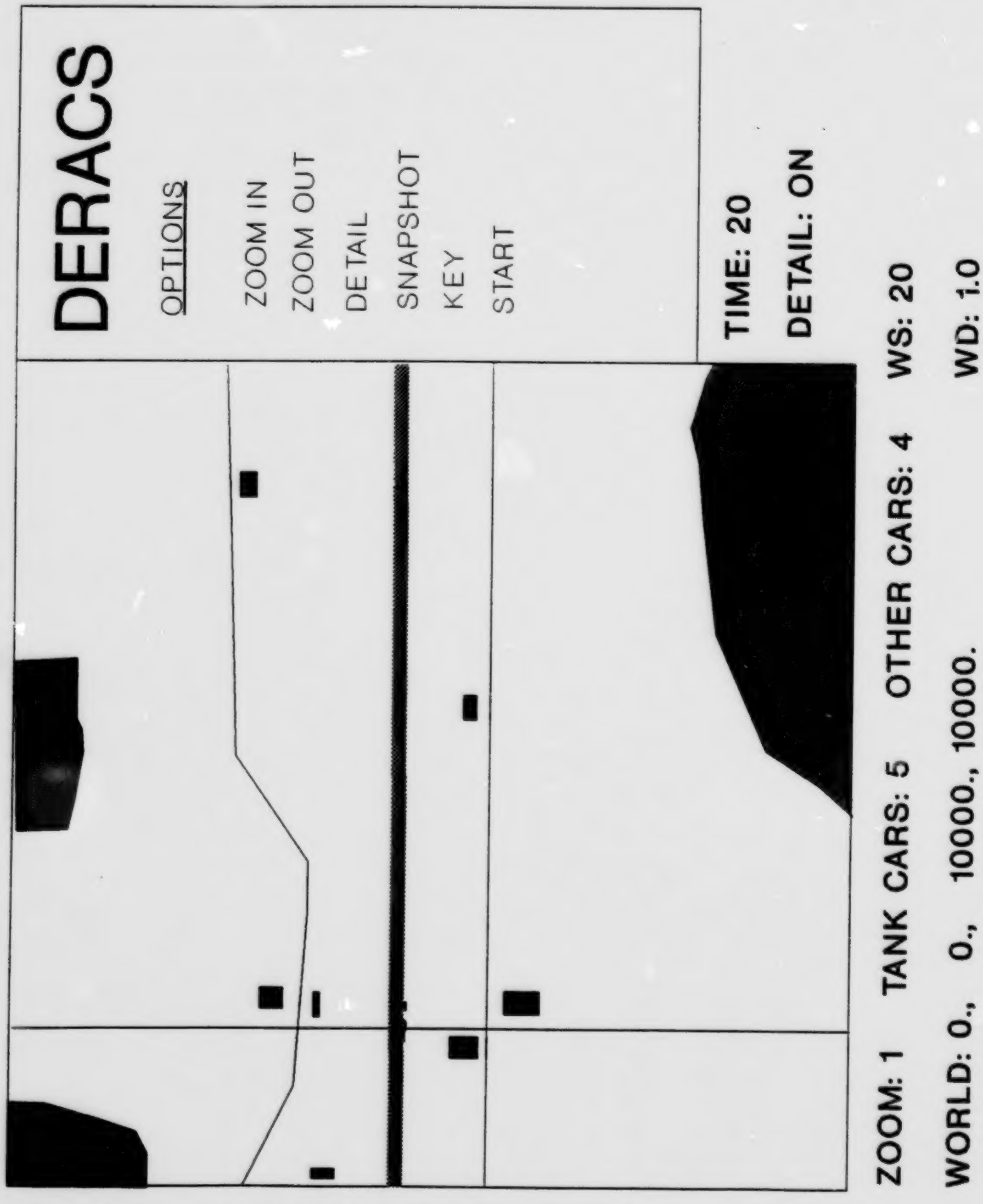
ZOOM OUT - see Section 4.3

ZOOM IN - see Section 4.3

DETAIL - This function allows you to unclutter a screen by removing events such as plumes, puffs and spills one at a time. When you select this command, a menu will appear displaying a list of events. As you select events, they will be set on or off. A negative selection will turn off the event and a positive selection will turn on the event. Reset will turn all events on. Upon exiting the command, the screen will be redrawn, showing only events that are turned on.

SNAPSHOT - This command prints out a picture of the screen on your epson or epson compatible printer. Before the scene is dumped, it is redrawn in white with a black background. When the printing is complete, the screen is redrawn in colour and you are returned to ready mode.

KEY - This command invokes the key page described in Chapter 4.


DERACS
OPTIONS
ZOOM IN
ZOOM OUT
DETAIL
SNAPSHOT
KEY
START

TIME: 20

DETAIL: ON

ZOOM: 1 TANK CARS: 5 OTHER CARS: 4 WS: 20

WORLD: 0., 0., 10000., 10000. WD: 1.0

FIGURE 6.1 SAMPLE POSTPROCESSOR OUTPUT

START - When this is chosen, the postprocessor starts reading and displaying the time steps at computer speed. The time for each display varies with the complexity of the drawing.

After it has drawn the next time step, it looks at the mouse and if either button is being depressed, the postprocessor stops and returns to ready mode.

If the mouse buttons are not being depressed, then the next time step is loaded and displayed.

EXIT - This function stops the postprocessor and dumps the Text Summary created up to that point. The Text Summary is printed on your epson or epson compatible printer and written to a report file.

6.4 TEXT REPORT SUMMARY

When exiting the postprocessor, a report is generated and written to a file with the name given by the user. This report contains two sections. The first section is a description of the parameters entered in the preprocessor including the train consist data. The second section is a summary of the events which happened during the postprocessor session. It should be noted that this section contains the last time step viewed on the screen but not later. The text summary is illustrated in Figure 6.2. The following is a description of the headings:

EVT# - Event number. Used as a reference for an event.

START, END - Start or end time of an event. Time = 0 refers to events that happen during the derailment. A Time = "-" refers to an event with no end time. For example, a rupture has a start time but no end time.

DIM1, DIM2 - These fields will be explained later as they differ for each event.

TC1...TC8 - These fields refer to the tank car #.

The following is a description of the DIM1 and DIM2 fields by event.

RUPTURE (RP)

DIM1 - 1.0 - impact rupture
2.0 - thermal rupture

DIM2 - If rupture is total loss of containment, no value will be written. If rupture is inward round then the diameter will appear. The dimension is in meters.

SPILL (SP)

DIM1 - Maximum radius of the spill in meters

DIM2 - Height of spill in meters

FIRE (FR)

DIM1 - Type of fire

DIM2 - Not used

BLAST(BL)

DIM1 - Level #3 of blast effect

DIM2 - Level #3 of thermal effect

FLARE(FL)

DIM1 - Length of flare in meters

DIM2 - Not used

PUFF(PU)

DIM1 - Maximum radius of the puff

DIM2 - Maximum distance downwind of the puff

```
Title  -  SAMPLE#1
_______________________________________________________________________________
DERACS: Derailment Accident Simulator - Summary
_______________________________________________________________________________

Wind Speed:    8.   Wind Direction:    1.   Barometric Pressure:    101325.
Air Temperature:   10.
Train Speed:   90 Km/Hr   Derailment Leader:     1   Truck Number:     1
_______________________________________________________________________________

Train Consist Data:

Consist # |  Veh. type  |  Comm. | Insul. | Init. Temp. |  Init. Fill

   1          1             -       -          -              -
   2          2             1       1         10.            90.
   3          2             2       0         20.            90.
   4          2             2       0         20.            90.
   5          3             -       -          -              -
   6          2             3       0         20.            90.

Title  -  SAMPLE#1
_______________________________________________________________________________
DERACS: Derailment Accident Simulator -
Summary  -  Global Time:     .3
_______________________________________________________________________________
Events Summary:

Evt#  Start  End     Dim1      Dim2     Tc1  Tc2  Tc3  Tc4

  1    .0      -        1.0        .3   RP   -    -    -     -    -    -    -
  2    .0      -        1.0      -      -   RP    -    -     -    -    -    -
  3    .0      -        1.0        .3   -    -   RP    -     -    -    -    -
  4    .0      -        1.0        .3   -    -    -   RP     -    -    -    -


  5    .0      -      249.2    401.0    -   BL    -    -     -    -    -    -


  6    .0       .0     20.6     120.6   PL   -    -    -     -    -    -    -
  7    .0       .0     14.2     54.2    -    -   PL    -     -    -    -    -

  8    .0       -      80.8     .02     -    -    -   SP     -    -    -    -

  9    .0       .0      1         -     FR   -    -    -     -    -    -    -
```

Figure C6.2 Sample Text Summary

PLUME(PL)

DIM1 - Maximum radius of the tip of the plume

DIM2 - Maximum distance downwind of the plume

6.5 RUNNING THE POSTPROCESSOR SEPARATELY

You can run the postprocessor independently of all other modules. An application for this is viewing data created during another DERACS session. To invoke the postprocessor, type ">post" from the DOS prompt. This will load the postprocessor, reorganize data files and start the session. Before typing ">post", be sure you have the correct data files in the \DERACS directory.

For details on these files, see Section 7.

7.0 FILE MAINTENANCE

Included with your DERACS files are two batch files which are used to store and retrieve DERACS data files. The storage utility is called STORE.BAT. To run this file, type ">STORE eee" at the dos prompt. The letters "eee" refer to the 3 letter file extension used when renaming the DERACS data files. When this program has been run, you can easily store the dat files using dos copy or some other backup utility. The retrieve utility will rename all the files changed by the store utility to the proper extension. To run this file, type ">RETRIEV eee" at the dos prompt. All the files will be renamed and you can then run the postprocessor.

Note: In both cases, the files to be STORED or RETRIEVED must be in the default directory.

When backing up the data files created by DERACS, be sure to have at least 7 high density floppy disks at hand. It is recommended that you use the program FASTBACK or something similar to backup data files.

APPENDIX D

SIMULATION SOFTWARE TECHNICAL DESCRIPTIONS

APPENDIX D

TECHNICAL DESCRIPTIONS OF PRIMARY SUBMODELS/SUBROUTINES

D.1 Program DERAIL

D.2 Subroutine BLAST

D.3 Subroutine BLEVE

D.4 Subroutine CONREL

D.5 Subroutine FIRBAL

D.6 Subroutine FIRIMP

D.7 Subroutine FLARE

D.8 Subroutine PFIRE

D.9 Subroutine SPILL

D.10 Subroutine TORIMP

D.11 Subroutine PLUME AND PUFF

APPENDIX D.1

PROGRAM DERAIL

1. INTRODUCTION

The derailment of a train can be a spectacular event, creating a great deal of noise and destruction, or it can be a relatively gentle occurrence where only a single truck leaves the rail without doing significant damage to either the vehicles or the surroundings. Even with a large-scale derailment, the local damage which can be done by a fast moving train is often insignificant compared to the far reaching results of fires which engulf derailed tankcars, possibly causing explosions and certainly contaminating the atmosphere with the by-products of combustion.

The subject of derailments and the damage that they may cause is of interest to many people in the Transportation field. Although it is not difficult to envision the sequence of events that take place during a derailment (cars leave the track, some of these are decoupled, some roll over, motion stops with cars distributed around the scene, some of these are tankcars, some tankcars are ruptured, some are on fire, ...), it is very difficult to obtain quantitative predictions of the consequences of the accident. With the aim of providing these quantitative results, this report forms one part of a derailment simulation which covers all aspects of the accident. In particular, the development of a dynamic model of the mechanics of train derailments is described here.

The equations-of-motion are presented, including inertial effects, the very significant forces of interaction between the cars and the ground and the forces acting between cars which must be borne by the couplers. Provision is made for the possibility of broken couplers and for subsequent damage to nearby cars caused by these couplers. Three types of car are included: locomotives, tankcars, and a generic freight car. The ability of decoupled cars to roll over is included in the model and provides a significant input to the simulation of burning tankcars.

Throughout this report, every effort is made to point out and justify assumptions and simplifications which were made during the development of the simulation. Many of these restrictions are forced by the use of modestly powered personal computers for the simulation. Others are dictated by a current lack of information regarding significant influences on the dynamic motion such as the true behavior of couplers and ground forces. Where possible, comparisons are made with the only other documented model of train derailments, that of Yang et al. [1,2]

In each of the following sections, the presentation will begin with the properties of the ideal model and proceed to describe the model which has been incorporated in the computer program which accompanies the report. A final section which summarizes the perceived weaknesses in this model and makes recommendations for future work is included.

---

(1)

2. THE MODEL

The details of the modelling of the derailment are included in the following sections. The description begins with the physical model and culminates with some discussion of the numerical techniques used for obtaining a solution to the equations-of-motion.

2.1 Describing the Motion (Degrees-of-Freedom)

The most important first choice in any dynamic simulation is that of choosing the variables which will be used to describe the motion of the system. These are the degrees-of-freedom of the system.

Ideally, a derailing train should be modelled as a set of three dimensional rigid bodies which are connected to each other by couplers and to the rails (or ground if the vehicle is no longer on the rails) via truck frames, wheelsets and at least two levels of suspension. Each body in the system would require three linear and three angular degrees-of-freedom to specify its location and orientation, leading to six variables per rigid body. Given a standard vehicle type having one carbody, two truck frames, and four wheelsets, it is a simple matter to calculate that a total of 42 degrees-of-freedom are required per vehicle. For a train of 40 cars there will be 1680 degrees-of- freedom, each requiring the derivation of a force or moment balance equation and leading to a second order differential equation to be solved. The numerical solution (discussed in Section 2.9) requires that the differential equations be transformed to first-order equations, resulting in a set of twice 1680 or 3360 differential equations to be solved simultaneously.

Although 3360 equations do not rule out the use of a personal computer on the basis of memory requirements, it does lead to an excessive amount of computer time to obtain a solution. Clearly, any simplification to the model which reduces the number of degrees-of-freedom without being detrimental to the accuracy will be desirable.

Two approaches to reducing the number of degrees-of-freedom have been taken. The first of these involves the use of constraints within the train. If wheelsets are assumed to remain fixed relative to the truck frames, then their motion is completely specified by the frame motion and individual cars in the train may be modelled as having only three bodies each (a carbody and two trucks, where the wheelsets are integral parts of the trucks). This reduces the required number of degrees-of-freedom to 18 per vehicle or 720 overall (1440 first-order differential equations). Although a significant reduction, this is still far too many for a reasonable implementation on even a very fast personal computer.

The assumption that wheelsets remain fixed to the trucks is not without error. There have been many reports of derailments where wheelsets have been found some distance from the site of the accident. However, the overall accuracy of the simulation will not be affected adversely by this approximation since the main concern is the location of the carbodies after the accident. The saving in complexity realized from the elimination of such a large number of degrees-of-freedom is warranted.

A further simplification is gained by also assuming that trucks remain coupled to the carbodies. This is a no more significant assumption than that made for the wheelsets since the trucks are no more massive and both the wheelsets and the trucks contribute primarily to the inertial properties of the vehicles. Naturally, the effects of free flying wheelsets and trucks cannot be simulated under these assumptions. This is not a serious restriction since the primary purpose of the derailment model is to predict the placement of tankcars subsequent to an accident.

Driven by the need to run the simulation on a small computer system, one final simplification has been made. The need to retain three-dimensional motion is seen as important for the prediction of carbody roll only. The other motions taking place in the vertical direction (i.e. heave and pitch) are not of importance. Heave motions will be constrained by the local topography so that the vehicle cannot finally lie anywhere but on the ground. Pitch motions are also constrained by the ground unless the vehicle turns end-over-end during the derailment. There is no evidence to suggest that this is a very likely occurrence and pitch has been removed from the model in order to reduce the computing load.

As a result of the simplification to two dimensional motion plus roll, as well as assuming that wheelsets remain coupled to trucks which remain coupled to carbodies, it is possible to describe the motion of each railcar with only 4 degrees-of- freedom. These are forward motion, lateral motion, yaw, and roll. As well, the model of roll motion uses a work-energy relationship described in Section 2.6, so that roll is found from an algebraic, rather than a differential, equation.

There remain only 3 dynamic degrees-of-freedom per vehicle after these simplifications, leading to 6 first-order differential equations per car or a total of 240 for the 40 car train used as a limiting size. The chore of solving this many simultaneous differential equations is not beyond the capabilities of personal computers. The actual computer time depends on the length of the simulation (i.e. elapsed time before all cars become stationary) as well as the number of cars in the train. A 40 car train derailing at 40 m/s can be expected to require significant computation time whereas a 10 car train at 20 m/s will require much less. The actual computing time is, of course, dependent on the computer system used.

The number of degrees-of-freedom is simple to calculate, but the choice of what physical significance each should have is not obvious. Since the position of the vehicle is of interest, it may be argued that the x,y coordinates of the vehicle centre of mass with respect to a fixed reference point and a yaw angle would be ideal. The problem with this system is that force balance equations must be written in directions which are not aligned with the vehicle axes. A better choice is to use a set of body-fixed axes which are standard for road vehicle analyses but are not usually applied to rail vehicles. Since the vehicles are leaving the rails in this simulation, the use of the body fixed axes is justified. This coordinate system is shown in Figure D.1-1. The primary dynamic variables are the forward speed, U, the lateral speed, V, and the yaw rate, r.

Force balance equations require that the acceleration of the vehicle be written in terms of these variables. The resulting longitudinal acceleration is:

$$a_{LONG} = \dot{U} - Vr \qquad (1)$$

The lateral acceleration is:

$$a_{LAT} = \dot{V} + Ur \qquad (2)$$

and the yaw acceleration, used in a moment balance equation, is simply, $\dot{r}$.

In addition to force balance equations, it is still necessary to determine the position of the vehicle. This is done by using the variables, x, y, and (yaw angle) shown on Figure D.1-1. By writing expressions for the velocity of the vehicle in terms of these variables, three additional differential equations arise. They are:

$$\dot{U} = \dot{x} - y\dot{\theta} \tag{3}$$

$$\dot{V} = \dot{y} + x\dot{\theta} \tag{4}$$

$$r = \dot{\theta} \tag{5}$$

Finally, the position of the vehicle with respect to a fixed reference position (point O) can be calculated from.

$$X_p = x \cos \theta - y \sin \theta \tag{6}$$

$$Y_p = x \sin \theta + y \cos \theta \tag{7}$$

2.2 Coupler Forces

Among the most significant forces acting during the derailment are those which act through the couplers to hold the vehicles together. In the past, couplers have been prone to disengage due to a relative vertical motion between neighboring cars. This problem has been largely cured through the use of "shelf couplers" which are designed to avoid just this problem. Variations of these couplers are now in almost universal use and the statistics show that couplers rarely separate in this manner today.

There are many papers on the subject of railcar couplers and draft-gear. Some of these are Phillips and Role [3], Ward and Leonard [4], Wallace [5], Raidt et al [6], and Wikander [7]. The draft-gear is essentially a cushioning device meant to absorb energy during impacts between cars. A typical curve of draft-gear resistance versus stroke shows very different stiffnesses for compression and release. The total travel allowed within the draft-gear is on the order of 0.06 m after which the forces must be absorbed by the elasticity of the coupler shanks. Wallace [5] gives an elastic limit for coupler shanks of roughly 2.3 million Newton.

An attractive way in which to include couplers in the model is through the use of constraint equations. That is, two cars which are coupled can have their combined two-dimensional motions described with four rather than six degrees-of-freedom. The three standard degrees-of-freedom (Figure D.1-1) can be used for one of the vehicles, with the motion of the other being totally specified by the position of the coupler (known from the first vehicle) plus a relative yaw angle about the coupler. In essence, a train of N cars needs only (N+2) degrees-of-freedom to completely describe the motions of all cars in the train. This results in 42 degrees-of-freedom rather than 120 for the 40 car train considered here.

The drawback to using this approach is that the coupler forces, calculated from the constraint equations, cannot be used to initiate a coupler failure no matter how large these forces are. The model relies on the cars remaining coupled since there are too few degrees-of-freedom to predict the motion of the train if a car does not have a point in common with both its neighbors. This is the basic assumption made by Yang et al. [1,2] who cannot predict decoupled behavior of rail cars. Even with this assumption, these authors claim success in predicting the numbers and placement of cars after an accident.

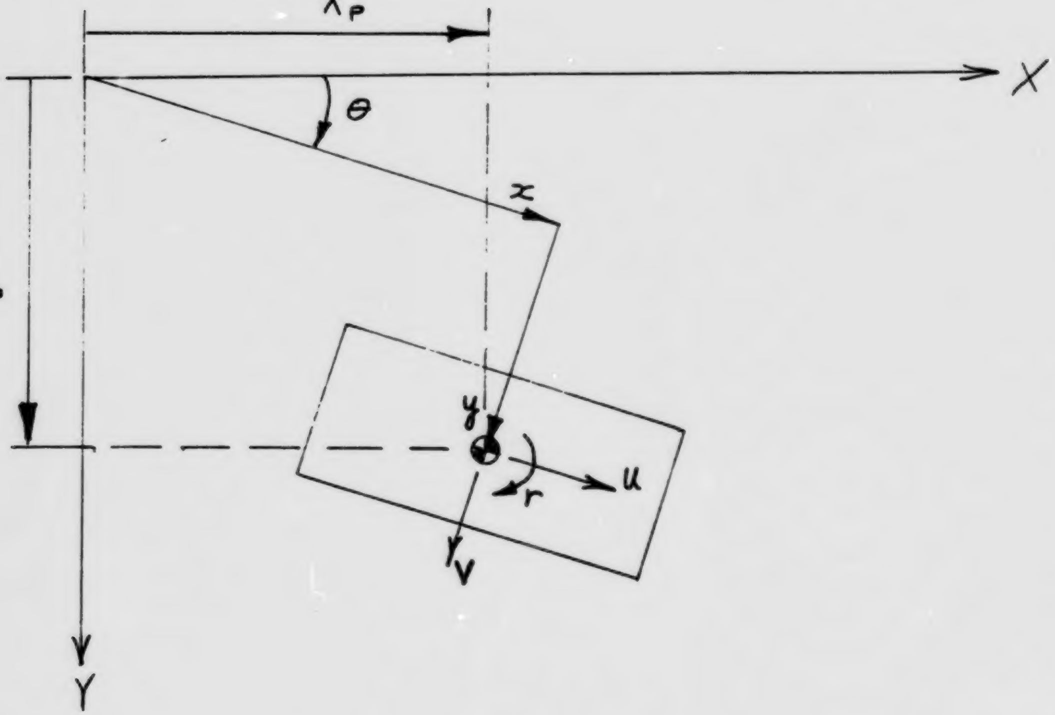

X_P
X
θ
x
Y_P
y
u
r
V
Y

Figure D.1-1 The Degrees-of-Freedom

Although it would be computationally efficient, there can be no further simplification of the degrees-of-freedom described in Section 2.1, if coupler separation is to be modelled. Therefore, the couplers are assumed to resist separation by providing a force related to the relative displacement between adjacent couplers (a stiffness).

The use of this simple relationship is meant to provide some idea of the force necessary to link the railcars together. It is, in fact, a simulation of the constraint force which must exist in order to maintain continuity of the train. There is no need to include a very accurate representation of the draft-gear forces since the small stroke allowed is quickly exceeded and the use of a stiff spring model is not inappropriate.

Figure D.1-2 gives some details of the geometry of this calculation. The force is given by:

$$F_c = K\Delta \tag{8}$$

where dx, dy and are shown on Figure 2.2 and,

$$\Delta = dx \cos \alpha + dy \sin \alpha \tag{9}$$

$$\tan\alpha = dx/dy \tag{10}$$

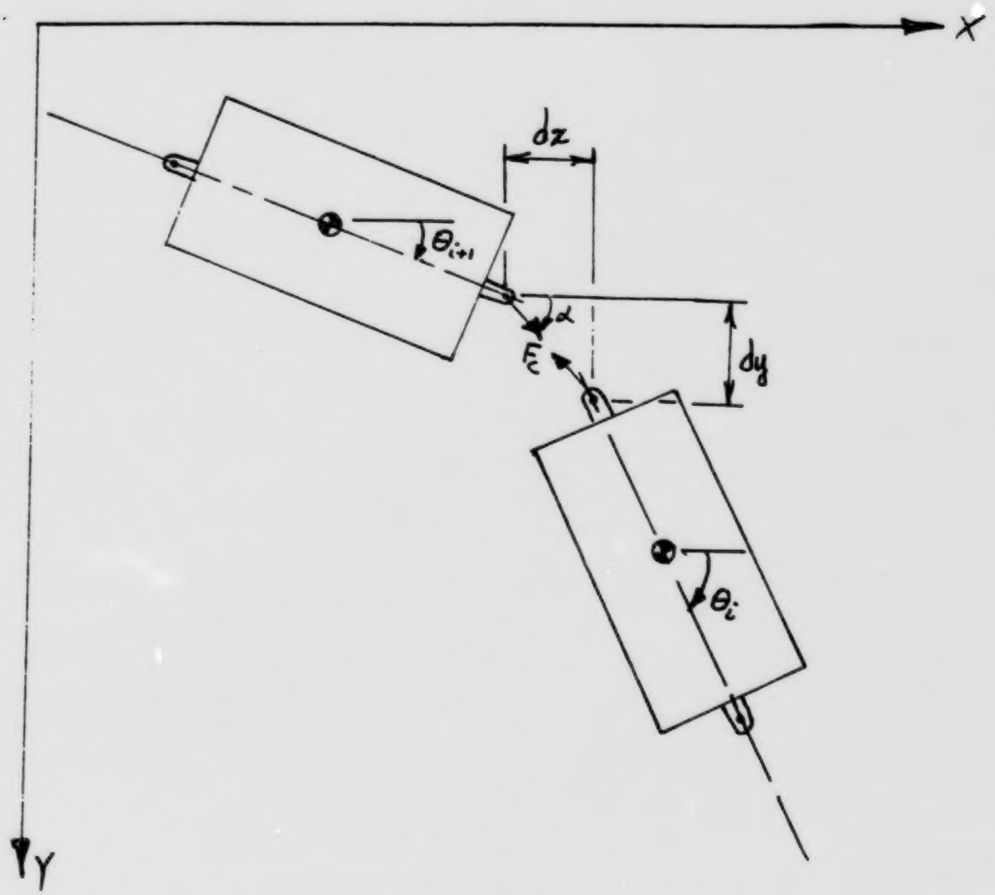
The couplers are assumed able to provide any required compressive load without failure but they are limited in their capability to sustain tensile loads. If the predicted coupler tensile force exceeds a preset maximum, the coupler is deemed to have failed and can carry no more load, effectively allowing the ends of two vehicles to move without restraint. A value for the limiting coupler force cannot be easily determined from available literature. Allowing for some plastic deformation, it will be somewhat larger than the elastic limit quoted by Wallace [5]. The coupler stiffness used is not an important parameter as long as it is large enough to eliminate large coupler deformations without breakage. The stiffness simply provides a mechanism for predicting the constraint forces and releasing them if they are large enough to cause breakage.

2.3 Ground Forces

The modelling of forces imparted to the vehicle by the ground follows closely other models of soil interaction forces. In almost all cases, it is assumed that a body sliding over the terrain is acted upon by a Coulomb type frictional force. That is, the magnitude of the force is equal to the normal force scaled by a coefficient of friction. The direction of the force is such that it acts in opposition to the velocity vector of the point in contact with the ground.

The effective point of application of the soil forces is not easily determined. If the car is upright and still retains its trucks, then there will be two forces, one at each truck centre. If the vehicle has lost its trucks or has rolled, then there will be a distribution of normal and frictional forces acting along the entire length of the vehicle. The net force acting then may still be approximately modelled by allowing for two frictional forces, one at each truck centre. Although not an exact representation, this is a computationally efficient assumption which provides resistance to both linear and angular motion as would be expected to be present.

The calculation procedure for ground forces is as follows. The velocity vector for each truck centre is calculated at each time step during the solution. The normal force at that point, approximated by one-half of the body weight, is combined with the ground coefficient of friction to give the force magnitude. The direction of this force is then specified to be in



X
Y
dz
dy
θ_{i+1}
θ_i
α
F_c

Figure D.1-2 Coupler Forces

the negative sense of the velocity vector and the force is divided into lateral and longitudinal components. These go directly into the force balance equations for the car. The lateral force also enters the car moment balance equation. Figure D.1-3 gives some of the details of this calculation.

2.4 The Equations-of-Motion

The equations-of-motion for a single vehicle can be written at this point in general terms. The following sections of the report fill in some of the details required for particular vehicles, depending upon their location, but the basic equations consist of two force balances and a moment balance.

Figure D.1-4 contains a free-body diagram of a representative car from the train, showing the soil interaction forces acting at the truck centres and the coupler forces at the ends of the vehicle. This figure is drawn for car N in the train where N is any number between one and the total number of cars.

The longitudinal force balance equation is:

$$M(\dot{U} - Vr) = F_{CL}^{x} - F_{L}^{x} - F_{T}^{x} - F_{CT}^{x} \tag{11}$$

The lateral force balance equation is:

$$M(\dot{V} + Ur) = F_{CL}^{y} - F_{L}^{y} - F_{T}^{y} - F_{CT}^{y} \tag{12}$$

and the moment balance in the yaw direction is given by:

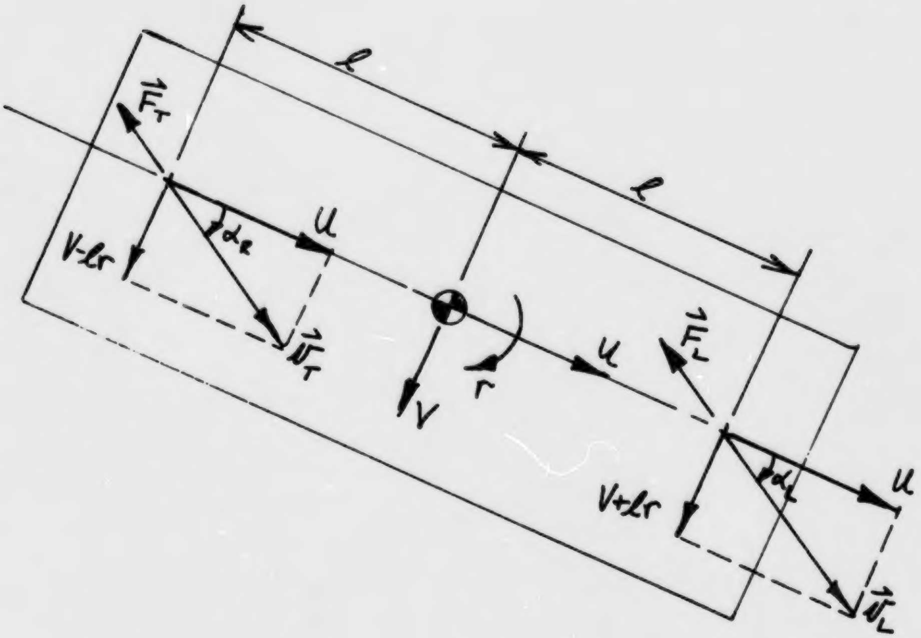
$$I\dot{r} = F_{CL}^{y}(a+l) - F_{L}^{y}(l) - F_{T}^{y}(l) - F_{CT}^{y}(a+l) \tag{13}$$

The magnitudes of the coupler and soil forces are calculated based on the descriptions of these forces given in the previous sections.

2.5 The Initiation of Derailment

The derailment is assumed to be initiated at a particular point on the rails where a truck derails. This point becomes the reference location for all subsequent calculations. It is assumed that there will be a break in the rails at this point and vehicles passing trough the reference point will no longer be guided by rails but will be subject to soil interaction forces.

All coordinates are referred to the derailment point using a coordinate system defined as follows and as shown in Figure D.1-5. The local tangent to the track is defined as the x-direction with positive coordinates coinciding with the direction of motion of the train. This allows cars to be easily classified as having passed the point of derailment (x greater than zero) or as not yet having reached this point (x less than zero). The direction normal to the local track tangent and pointing toward the high rail is defined as the positive y-direction. Railcars travelling around a curve when they derail will naturally move in the positive y-direction.



$\vec{F}_T$
l
l
α_t
u
$V-lr$
$\vec{v}_T$
V
r
u
$\vec{F}_L$
α_L
u
$V+lr$
$\vec{v}_L$

Figure D.1-3 Vehicle/Ground Interaction Forces

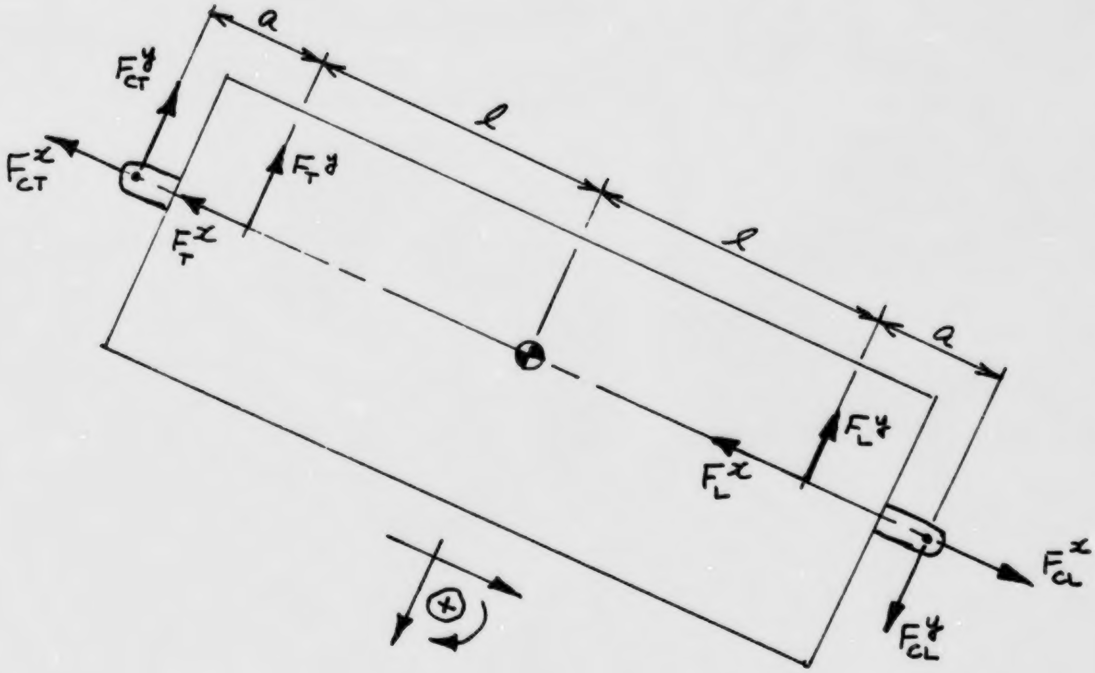

a
l
l
a
F_{CT}^{y}
F_{CT}^{x}
F_{T}^{y}
F_{T}^{x}
F_{L}^{y}
F_{L}^{x}
F_{CL}^{x}
F_{CL}^{y}
+

Figure D.1-4 Free-Body Diagram of a Car

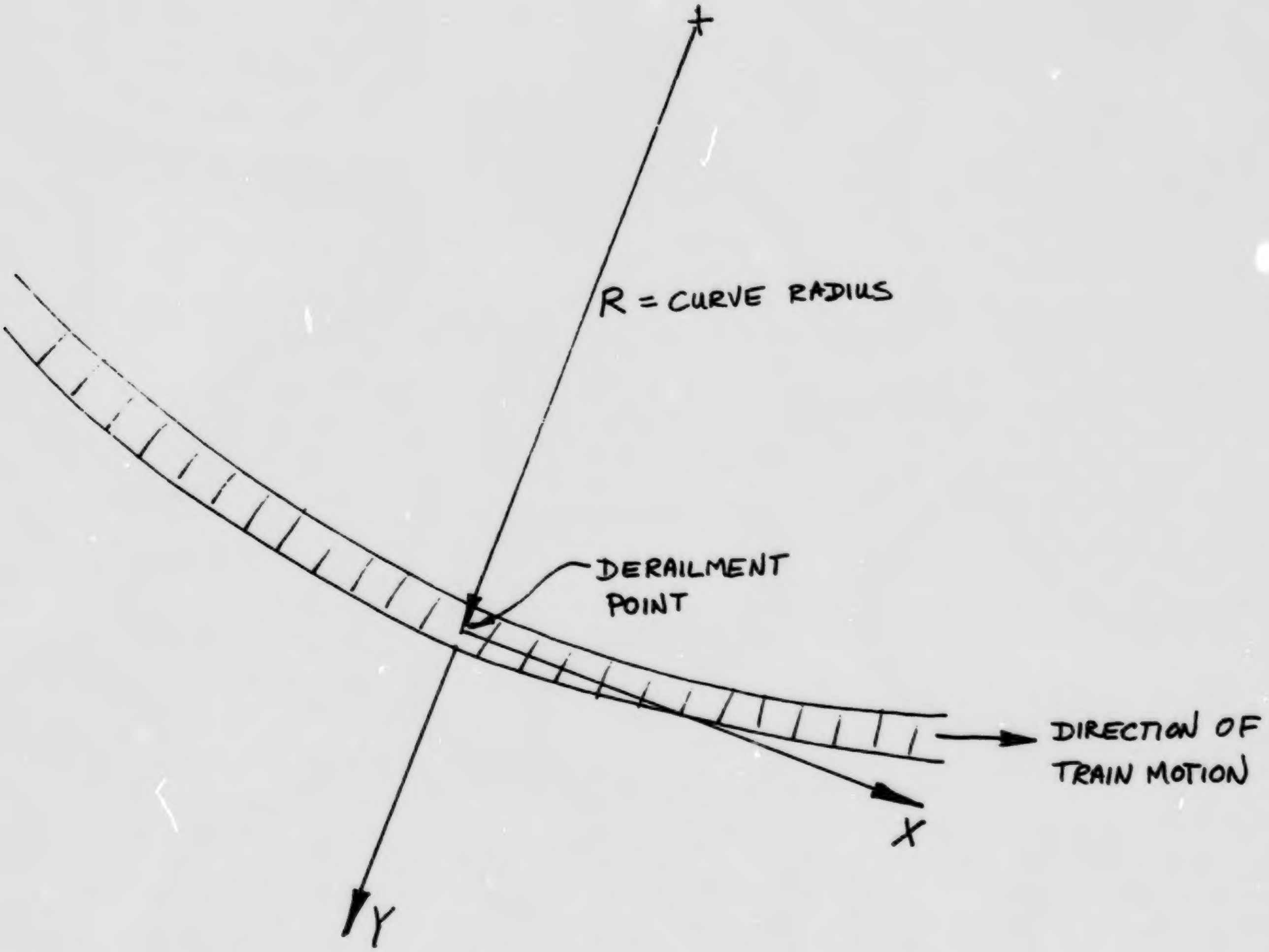

R = CURVE RADIUS
DERAILMENT POINT
DIRECTION OF TRAIN MOTION
X
Y

Figure D.1-5 The Derailment Coordinate System

If a train derails on tangent track, then the preferred direction of derailment is unknown and depends strictly on local conditions at the point of derailment. For tangent track accidents, the positive y-direction is arbitrary and simulated results may be reflected about the x-axis without loss of generality.

All positional output from the computer program is quoted with respect to the reference frame defined here.

There are three different regimes of interest during the derailment simulation. Separate formulations of the equations- of-motion are necessary for cars which were ahead of the derailment point when the accident occurred, for those which are derailed, and for those which have not yet reached the point of derailment. These different conditions are considered in the following.

2.5.1 Cars on the Rail Ahead of the Derailment

Cars which had already passed the derailment point when the accident occurred will be guided by intact rails. They will therefore not be subject to soil forces but instead to rail guidance forces. There has been a great deal of work done on the dynamics of vehicles on tracks but the complexity of including very accurate simulations of the leading vehicles is not warranted for the purposes of this work.

It is expeditious here to specify the lateral and yaw velocities and accelerations of these vehicles instead of the forces which cause them. Under the assumption that the vehicles will remain on the track, there will be a set of wheel/rail forces which cause no net lateral or yaw accelerations. These variables are therefore set to zero at each time step in the solution.

In the local x-direction (i.e. along the track) it is assumed that full emergency braking is being applied because of the derailment situation and that the coupler forces are acting in this direction. These vehicles will slow down and eventually stop due to these forces. There is also the potential to break couplers on these vehicles. Yang et al. [1,2] assumed that the leading coupler of the first derailed car always became disengaged so that the leading portion of the train had no further involvement with the derailed cars. That assumption is not necessary here. The leading part of the train is involved until one of the couplers breaks.

2.5.2 Cars in the Derailment Zone

The derailment zone is a loosely defined area because of the difficulty inherent in trying to predict the actual extent of track damage. Some assumptions are necessary here and have been made.

Initially, there is a derailment point as opposed to a derailment zone. A single truck is assumed to have come off the track at this point. There will undoubtedly be track damage in all but the most gentle of derailments so that the point becomes a zone. There are no grounds available for estimating the actual extent of the track damage so the assumption is made that the derailed car and its immediate neighbors both fore and aft are guided, not by rails, but by frictional ground forces. These forces are not unlike wheel/rail forces in many respects and this broad assumption allows for simulation of both major and minor accidents.

As the accident progresses, the derailment zone moves forward with the car immediately leading the first derailed car. Track structure in the zone is assumed to contribute to the frictional resistance of the soil without providing the guiding forces for which rails are designed. In essence, the track will be destroyed by the sliding vehicles.

Any vehicle passing the point where the trailing neighbor of the first derailed car was located at the initiation of the derailment is assumed to be in the derailment zone and thus guided by the soil frictional forces.

This derailment scenario allows the propagation of the accident in both forward and reverse directions along the track. If a coupler breaks on the leading part of the train, it will continue freely, affected only by braking forces. The trailing part of the train will continue to pass into the derailment zone until motion comes to a stop.

2.5.3 Cars on the Rail Behind the Derailment

These are cars which have not yet entered the derailment zone. They are subject to forces similar to those for the leading cars discussed in Section 2.5.1. Emergency braking is applied and coupler forces act in the longitudinal direction. Unlike the leading cars, these may not stay in the same regime throughout the accident. Depending on the severity of the derailment, many of them will pass into the derailment zone and be treated as derailed cars as described in Section 2.5.2.

2.6 Tankcar Roll

The treatment of roll of the cars is a difficult problem. The roll angle cannot be considered to be a smoothly varying degree-of-freedom during the derailment like the lateral and longitudinal displacements and the yaw angle. These three variables respond to complex forces but are not subject to a threshold level below which they do not exist and above which they vary rapidly. Roll angle, on the other hand, has these properties.

An upright vehicle is supported by it wheelsets. The distance between wheels on any axle gives the vehicle breadth and this breadth resists any tendency of the vehicle to roll over, up to a limit. Once this limit has been reached, the vehicle rises up, supported by the wheels on one side only, and then may or may not tip over depending on whether the disturbing forces are capable of moving the centre of mass of the vehicle to a point on the other side of the supporting wheels. Once this has occurred, the vehicle will continue to roll under its own weight.

The analysis of roll is important to the prediction of tankcar burning and must be addressed in this model. Considering tankcars only, and ignoring the roll of other vehicle types, simplifies the problem to a certain extent. The analysis proceeds as follows.

Figure D.1-6 shows an end view of an upright tankcar with the lateral components of the forces acting on the car at the ground support points and at the couplers. This car is assumed to have left the rails so that it has a lateral acceleration which is shown on the diagram. Assuming that the car has not started to roll, a moment balance about the centre of mass in the roll direction results in the following static equation:

$$-F_c^y h_c - F_G^y h + N_2 d - N_1 d = 0 \qquad (14)$$

This equation contains the unknown normal forces acting at the wheels on the left and right sides of the vehicle. These normal forces must also support the weight of the vehicle, giving:

$$N_1 + N_2 - W = 0 \tag{15}$$

Equations (14) and (15) will allow the calculation of the normal forces on the left and right sides of the vehicle for any known set of soil and coupler forces. Since these forces are available in the simulation program, it is not difficult to calculate the normal forces. Rollover is impending when either the left or right normal force is reduced to zero.

There is another practical consideration which cannot be ignored. The coupler forces are adaptable in the sense that they respond to a relative motion between neighboring cars by providing a force designed to constrain the couplers to not move apart. If a rollover occurs without a decoupling, then either a sequence of connected cars must roll together or a single car must lose contact with the ground and roll about the coupler. Coupler design will not permit the second of these options to occur and the first is an unlikely occurrence. Given these considerations, this model permits rollover only for vehicles which are completely decoupled from their neighbors.

This restriction simplifies Equation (14) by removing all need to consider coupler forces in the roll analysis. Combining Equation (14) with (15) yields expressions for the normal forces in terms of the ground forces and the weight of the car. Rollover is imminent if either of the normal forces goes to zero. The direction of roll can be inferred from the side which lifts off the ground. Setting the normal forces to zero allows the limiting soil force to be calculated. For total ground frictional forces outside the range:

$$-\frac{dW}{h} < F_G^{\,y} < \frac{dW}{h} \tag{16}$$

rollover is initiated. In this expression, W is the tankcar weight, d is one half the track gauge (wheel separation) and h is the height of the tankcar centre of mass above the ground.

The computer simulation checks the range given in Equation (16) for all tankcars which have been totally decoupled during the accident. It is assumed that two things happen if a rollover begins. Firstly, unlike other cars in the simulation, the tankcars are assumed to lose their trucks. This is a realistic assumption in most cases and allows the use of a relatively straight forward energy method for calculating the resultant roll angle.

Once the trucks have left the tankcar, it becomes a rolling and sliding cylinder which is moving under the influence of non- conservative (i.e. frictional forces) acting at the contact between the car and the ground. The initial kinetic energy of the car, which can be calculated from its velocity just as rollover begins, will be totally expended by the time motion has stopped. The work done by the frictional forces during this interval will be equal to the initial kinetic energy. Some work will be necessary to halt the linear and yaw motions of the car. This amount is easily found by considering the total linear motion after rollover which is calculated via the equations-of- motion discussed in earlier sections of this report. The remainder of the energy must be expended by work done during rolling motions. Figure D.1-7 shows the frictional force acting on a rolling tankcar. The resulting expression for the final roll angle is:

$$\phi = \frac{S}{R} - \frac{V^2}{2\mu g R} \tag{17}$$

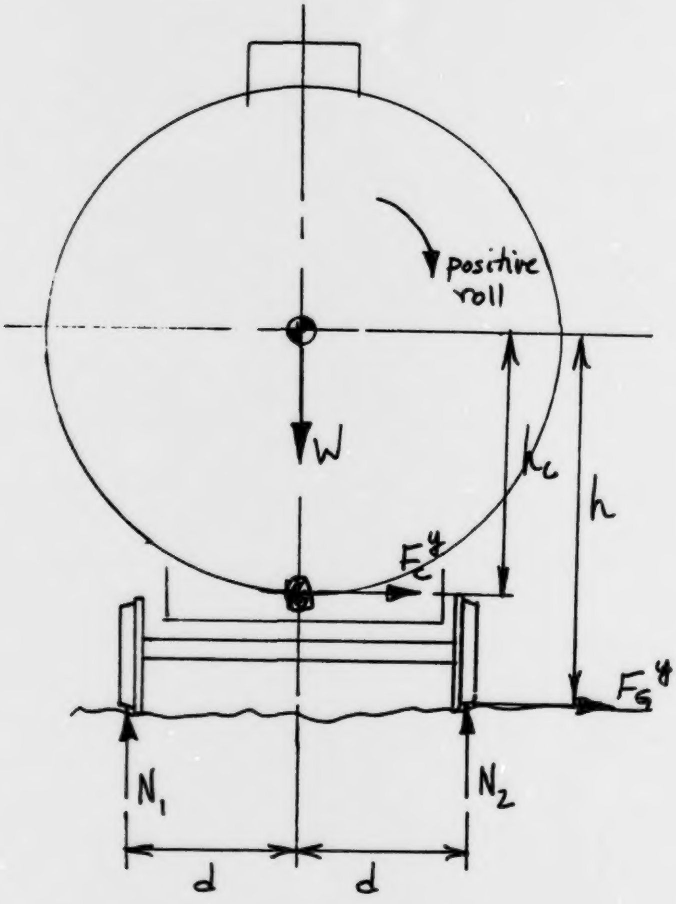

positive roll
W
h_c
h
F_c^y
F_G^y
N_1
N_2
d
d

Figure D.1-6 End View of an Upright Tanker

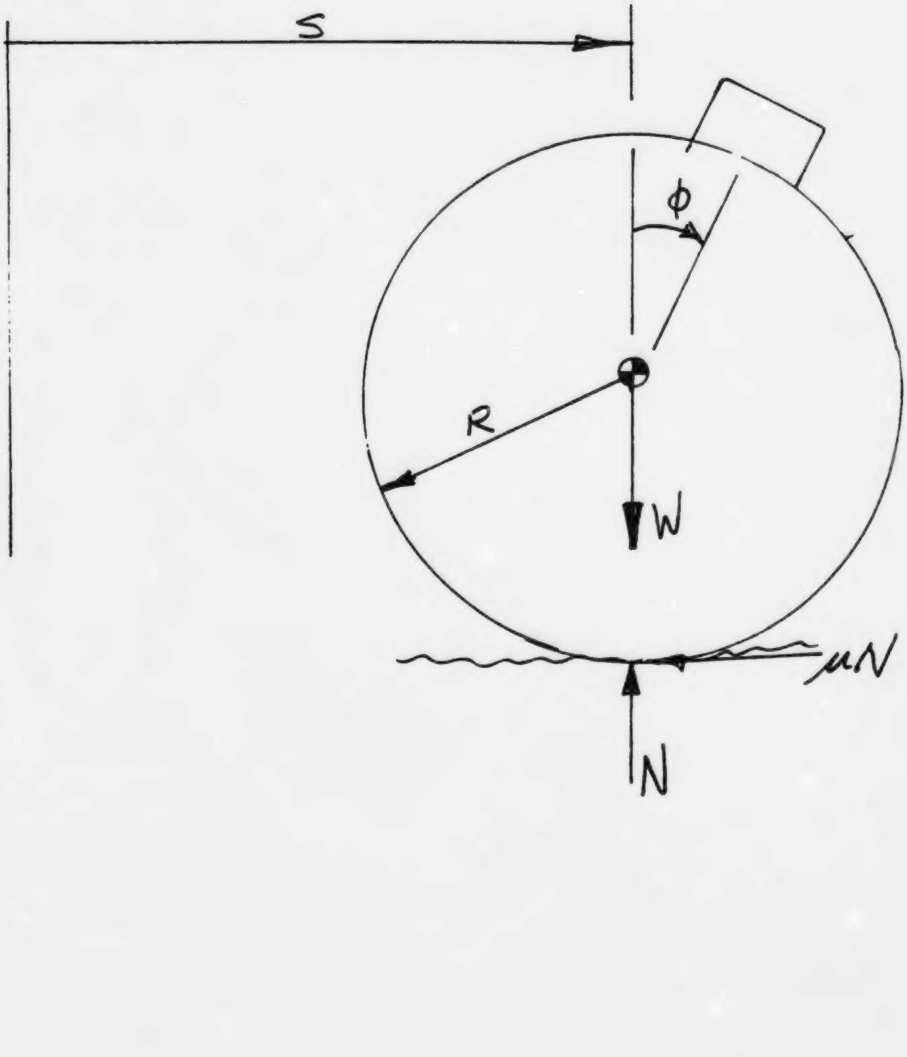


Figure D.1-7 The Rolling Tankcar

There is one other consideration here. Tankcars generally have a dome at their top which is an impediment to rolling. The simulation uses the energy method up to the point where the dome has contacted the soil (roughly 120 degrees of roll) and then stops any further roll.

2.7 Tankcar Ruptures

Another important consequence of derailments is the potential rupture of tankcar walls through contact of the car with other derailing cars. Accurate prediction of this phenomenon would require a model of the impact forces acting between two cars which come into contact during the accident. These forces, which are dependent upon the relative velocity of impact, the shapes of the contacting parts and material properties, would then be compared to the strength of the material and ruptures could be predicted.

In the model described here, a simpler approach is made necessary by restricted computer time and a need to simplify the input description of the train so that useful results can be predicted without excessive attention to detail. As a result, it is assumed here that tankcars will only be punctured by free couplers on other cars. The case of two cars colliding side-to-side will not result in a rupture. If a free coupler comes into contact with a tankcar, the tankcar is deemed to have been ruptured. If the relative velocity of the cars is primarily towards each other, the puncture will consist of a relatively circular hole having a diameter larger than that of the coupler. Typical measurements of coupler punctures indicate that a diameter of 0.75 m is typical. If there is a significant sliding velocity of the tankcar relative to the impacting coupler, then the rupture is categorized as a long tear in the tank leading to total loss of contents.

The simulation output for all tankcars includes a description of the rupture if it exists. This description is limited to the position and diameter of clean punctures or an indication that the contents have been completely lost.

The positions of all couplers, whether broken or intact, are calculated at each time step of the solution. It is a relatively simple matter to check all broken couplers for intrusion into a zone around each tankcar in the train. Such intrusions will be noted and classified as ruptures.

2.8 Effect of Track Curvature

The effect of track curvature on the derailment is incorporated in the model. Although it is common to think of vehicles travelling around corners at high speed as being subject to large forces which are attempting to throw the vehicle off the curve, this is not the case. The cars would actually prefer to move in a straight line but are forced to go around the curve by rail forces directed toward the centre of the circular path they are following. The fact that a car has derailed simply means that these rail forces no longer exist and that the car is free to follow the straight path it desires insofar as it is permitted to do this by the ground on which it travels.

The cars which are still on the rails will follow the curve and there will be an attempt by the couplers to maintain contact with the derailed cars. This will only be successful so long as there is no coupler breakage.

From the point of view of the equations-of-motion there is no change to the formulation presented in the preceding. The simulation views the track as moving away from the derailed cars as they progress across the ground. The final result is that the cars will be distributed farther away from the track structure, but they essentially follow a straight trajectory once they are past the derailment point.

All simulation output positions are quoted in terms of the reference frame described in Section 2.5 and shown in Figure D.1-5. Any graphical display showing the cars distributed far out on the high rail side shows the track curving away from the x-axis which can be thought of as the line of derailment.

2.9 The Numerical Procedure

Techniques for solving ordinary differential equations on digital computers are widely available and many different approaches may be taken. There are two tasks required of the person developing the simulation. That person must first derive the equations-of-motion and cast them in the form of a set of first-order ordinary differential equations. This process has been described in the material preceding this section. The second task is that of choosing a suitable solution routine.

The requirements for this project were that the simulation should be able to run on a personal computer in a reasonable time. There is also the implicit requirement that the solution accuracy be as high as possible. The inaccuracies caused by approximations and simplifications made in the derivation of the equations-of-motion can be insignificant compared to those caused by a poor solution routine.

There is unfortunately a trade off between solution speed and accuracy. A very low order solution (e.g. fixed step Euler's Method) will result in very fast solutions with errors which are indeterminate and may or may not be large. This simulation initially employed the Euler technique, as does that of Yang et al. [1,2], in an effort to reduce computing time. Another program employing the Runge-Kutta Fehlberg solution with variable time step to control error was produced at the same time.

Comparisons between the simulated results clearly indicated that the Euler method was incapable of producing acceptable solutions. The program therefore uses the variable time-step Runge-Kutta Fehlberg technique. This provides consistently accurate solutions within an error tolerance set by the user.

3. WEAKNESSES IN THE MODEL

There have been many assumptions and simplifications made in the development of this model of train derailments. On a positive note, this model has eliminated a number of weaknesses found in the model of Yang et al. [1,2]. The inclusion of decoupling, tankcar roll, tankcar puncture, and improvement of the solution technique number among these.

There also remain a number of weak areas which deserve further investigation. In reading this report critically, these will be quickly spotted. In particular, the following problem areas need further work.

The prediction of roll angle is of vital importance to subsequent simulations of burning tankcars. A fully three- dimensional vehicle model, including roll as a dynamic variable, should be developed to improve this prediction.

The use of three-dimensional models also permit heave and pitch to be included. Here, they were quickly dismissed in an effort to reduce computing effort. They may be important and should be included in future models.

There is a need to increase the complexity of the input description of the derailment. Many important topographical features such as trackside grade and obstructions will affect the results. These have been completely ignored in this model.

The tank puncture mechanism needs to be addressed. The assumption of puncture whenever a free coupler contacts a tankcar is certainly not ideal but was necessary here. Along with this should come an improved model of the vehicle collisions which will allow the contact forces to be modelled. These forces, though important to the dynamics, have been left out for now.

Lastly, there is a definite need to use more powerful computers. None of the items listed here can be efficiently implemented on personal computers. The potential to use fast, high resolution graphics to display the results will also come with the more powerful computers.

4. PROGRAM USER'S GUIDE

The model described above has been implemented in a FORTRAN computer program named DERAIL. The source code for this program is included as Section 5 of this report. This section is a user's guide for the software.

4.1 Input Data

In line with the attempt to simplify as much of the analysis as possible, there is only one input file required and it is shared with the burning tankcar simulation. The user does not have the freedom to change many system parameters from the data file. Those which are not controlled from the input file are built into the computer code and, if necessary, may be changed there.

The input data is contained in a file called CARDATA.DAT. Free format READ statements are used so that data type and placement in the record are not critical. The structure of this file is as follows:

RECORD 1 : 2 real numbers, 2 integers

VARIABLE NAMES : VEL,RADIUS,ICDRL,ITRUCK

VEL = train forward speed (metres/sec)
RADIUS = curve radius at the derailment point (metres)
ICDRL = number of the derailed car. Cars are numbered rearward from the leading car to the final car.
ITRUCK = +1 if the leading truck of car #ICDRL derails
-1 if the trailing truck of car #ICDRL derails

RECORD 2 : 1 integer, others

VARIABLE NAMES : NCARS, other variables used by the burning tankcar program

NCARS = number of cars in the train (maximum is 40)

Note: Records 3 and 4 are repeated NCARS times.

RECORD 3 : 2 integers

VARIABLE NAMES : ICAR,ITYPE

ICAR = car number
ITYPE = car type
ITYPE = 1 - locomotive
ITYPE = 2 - tankcar
ITYPE = 3 - generic freight car

RECORD 4 : 1 integer, others

VARIABLE NAMES : ICOMM, other variables used by the burning tankcar program

Note: none of the variables in Record 4 is used in the derailment simulation. This record appears only if ITYPE = 2. A dummy READ statement is provided if Record 4 exists.

4.2 Output Data

The predictions of the derailment simulation are contained in two data files. One of these (DERAIL.DAT) provides input for the burning tankcar program. The other file (ANIMATE.DAT) saves the sequence of positions of the cars in even increments of time during the accident.

DERAIL.DAT contains one record for each car in the train with an additional record if the car is a tankcar. The data is presented by car number from the leading car to the trailing car. The structure is as follows:

Note: there are NCARS repeats of these records.

RECORD 1 : 1 integer, 4 real numbers

VARIABLE NAMES : ICAR,XP,YP,YAW,ROLL

ICAR = car number
XP = x-coordinate of the geometric centre of the car in the derailment coordinate system (Section 2.5, Figure 2.1). (metres)
YP = y-coordinate of the geometric centre of the car in the derailment coordinate system (metres).
YAW = Yaw angle of the car with respect to the positive x-direction of the derailment coordinate system. (radian)
ROLL = Roll angle of the car about the positive x-direction. (radian). This will be non-zero for tankcars only.

Note: Record 2 is printed for tankcars only. It is omitted for other cars.

RECORD 2 : 1 integer, 3 real numbers

VARIABLE NAMES : RUPTYP,RUPTD,RUPTH,RUPTL

RUPTYP = Rupture type
RUPTYP = 0 - no rupture
RUPTYP = 1 - total loss of contents
RUPTYP = 2 - inward, round puncture
RUPTD = Rupture diameter (metres). (RUPTYP = 2 only)
RUPTH = Rupture vertical position from ground (metres). (RUPTYP = 2 only)

RUPTL = Rupture longitudinal position measured from car geometric centre towards front of car (metres). (RUPTYP = 2 only)

The file ANIMATE.DAT contains the positions of all cars at each 0.1 second interval of the accident. The following records are repeated until the accident is over. These data may be used in conjunction with computer graphics to animate the occurrence.

RECORD 1 : 1 real number, 1 integer

VARIABLE NAME : TIME,NCARS

TIME = Time (seconds) since the initiation of the derailment.
NCARS = Number of cars in the train.

Note: Record 2 is repeated NCARS times

RECORD 2 : 1 integer, 4 real numbers

VARIABLE NAMES : ICAR,XP,YP,YAW,ROLL

ICAR = car number
XP = x-coordinate of the geometric centre of the car in the derailment coordinate system (Section 2.5, Figure 2.1). (metres)

YP = y-coordinate of the geometric centre of the car in the derailment coordinate system (metres).
YAW = Yaw angle of the car with respect to the positive x-direction of the derailment coordinate system. (radian)
ROLL = Roll angle of the car about the positive x-direction. (radian). This will be non-zero for tankcars only.

5. SOURCE CODE FOR PROGRAM "DERAIL"

```
      PROGRAM DERAIL
      IMPLICIT REAL*8 (A-H,O-Z)
C
C     TRAIN DERAILMENT SIMULATION
C
C     INCLUDES:
C
C     - UP TO 40 CARS ON STRAIGHT OR CURVED TRACK
C     - LOCOMOTIVES, TANK CARS, AND FREIGHT CARS
C     - COUPLER SEPARATION
C     - ROLL OF UNCOUPLED CARS
C     - TANKCAR RUPTURES FROM COUPLER IMPACT
C
      REAL*8 X(240),DXDT(240),MASS(3),INER(3)
      INTEGER RUPTYP
      COMMON/PARAM/ CARM(40),CARI(40),ICDRL,NCARS,EL,AD,ANGV,ITYPE(40)
      COMMON/COUPLR/ FCOUP(39),ALPHA(39),ICOUP(39)
      COMMON/POS/ XP(40),YP(40),YAW(40),ROLL(40),XCL(40),YCL(40),
     1 XCT(40),YCT(40)
      COMMON/RUPT/ RUPTYP(40),RUPTD(40),RUPTH(40),RUPTL(40)
      COMMON/INIT/ DRLST,IDRL(40)
      DATA MASS/1.8E5,1.2E5,1.2E5/
      DATA INER/4.5E6,3.E6,3.E6/
      DATA ELL/7./,ADIM/2./,IDIR/1/
C
C     OPEN I/O FILES
C
      OPEN(1,FILE='CARDATA.DAT',STATUS='OLD')
      OPEN(2,FILE='DERAIL.DAT',STATUS='NEW')
      OPEN(3,FILE='ANIMATE.DAT',STATUS='NEW')
C
C     READ CAR DATA
C
      READ(1,*) VEL,RADIUS,ICDRL,ITRUCK
      READ(1,*) NCARS
      DO 10 I=1,NCARS
      READ(1,*) ICAR,ITYPE(ICAR)
      CARM(ICAR)=MASS(ITYPE(ICAR))
      CARI(ICAR)=INER(ITYPE(ICAR))
      IF(ITYPE(ICAR).EQ.2) READ(1,*) IDUM
10    CONTINUE
C
C     SET CONSTANTS AND I.C.'S
C
      ANGV=0.2
      H=0.3
      NDOF=3*NCARS
      NEQN=2*NDOF
      EL=ELL
      AD=ADIM
      DRLST=AD+EL
      CL=2*(AD+EL)
      DO 15 I=1,39
15    ICOUP(I)=1
      ICNT=-6
      STX=CL*(ICDRL-1)
      DO 20 I=1,NCARS
      IDRL(I)=0
      ROLL(I)=0.D0
      YAW(I)=0.0
      RUPTYP(I)=0
      RUPTD(I)=0.0
      RUPTH(I)=0.0
      RUPTL(I)=0.0
      ICNT=ICNT+6
      X(ICNT+1)=STX-(I-1)*CL
      X(ICNT+2)=0.D0
      X(ICNT+3)=0.D0
      X(ICNT+4)=VEL
      X(ICNT+5)=0.D0
      X(ICNT+6)=0.D0
20    CONTINUE
C
C     DERAILED CAR GETS AN INITIAL ANGULAR DISPLACEMENT
C     POSITIVE IF LEADING TRUCK DERAILS
C     NEGATIVE IF TRAILING TRUCK DERAILS
C
      ICNT=6*(ICDRL-1)
      X(ICNT+6)=ITRUCK*ANGV
      ICNT=6*(ICDRL-ITRUCK-1)
      X(ICNT+6)=-ITRUCK*ANGV
C
C.....BEGIN SOLUTION
C
      HSTART=1.D-3
      HMIN=1.D-8
      HMAX=H
      TI=0.0
      TF=TI+H
      TOL=0.1
30    CONTINUE
      CALL RKFINT(TI,TF,X,DXDT,240,TOL,HSTART,HMIN,HMAX,IFLAG)
      IF(IFLAG.EQ.0) GO TO 40
      WRITE(2,9000)
9000  FORMAT(' ',//,3X,'*** RKFINT COULD NOT MEET TOLERANCE ***')
      STOP
40    CONTINUE
C
C     INITIAL CONDITIONS FOR RECENT DERAILMENTS
C
      DO 42 JCAR=ICDRL,NCARS
      IF(IDRL(JCAR).NE.2) GO TO 42
      IDIR=-IDIR
      IDRL(JCAR)=1
      ICNT=6*(JCAR-1)
      X(ICNT+6)=ANGV*IDIR
      GO TO 44
42    CONTINUE
44    CONTINUE
C
C     ADD CURVE RADIUS EFFECT BEFORE WRITING
C
      DO 45 I=1,NCARS
      IF (XP(I).LT.0.0) GO TO 45
      ANG=DASIN(XP(I)/RADIUS)
      YP(I)=YP(I)+RADIUS*(1.0-DCOS(ANG))
45    CONTINUE
C
C     WRITE TO ANIMATE.DAT
C
      WRITE(3,1000) TI,NCARS
1000  FORMAT(' ',F4.1,I4)
      DO 47 I=1,NCARS
47    WRITE(3,1100) I,XP(I),YP(I),YAW(I),ROLL(I)
1100  FORMAT(' ',I4,4E12.4)
      TI=TF
      TF=TI+H
C
C     CHECK FOR END OF SIGNIFICANT MOTION
C
      VCHK=0.0
      DO 75 ICHK=ICDRL,NCARS
      JCHK=6*(ICHK-1)
      UCAR=X(JCHK+4)
      VCAR=X(JCHK+5)
      RCAR=X(JCHK+6)
      VCHK=DSQRT(UCAR*UCAR+VCAR*VCAR+RCAR*RCAR)
      IF (VCHK.GT.3.0.AND.TI.LT.40.) GO TO 30
75    CONTINUE
C
999   CONTINUE
C
C     WRITE TO DERAIL.DAT
C
      DO 60 I=1,NCARS
      WRITE(2,2000) I,XP(I),YP(I),YAW(I),ROLL(I)
      IF(ITYPE(I).EQ.2) WRITE(2,2100) RUPTYP(I),RUPTD(I),RUPTH(I),
     1 RUPTL(I)
60    CONTINUE
2000  FORMAT(' ',I4,4E12.4)
2100  FORMAT(' ',I4,3E12.4)
C
      STOP
      END
C
C
```

```
C
C
      SUBROUTINE XDOT(N,TIME,X,DXDT)
      IMPLICIT REAL*8 (A-H,O-Z)
C
C     IMPLEMENTS THE EQUATIONS-OF-MOTION
C
      REAL*8 X(240),DXDT(240)
      INTEGER RUPTYP
      COMMON/PARAM/ CARM(40),CARI(40),ICDRL,NCARS,EL,AD,ANGV,ITYPE(40)
      COMMON/COUPLR/ FCOUP(39),ALPHA(39),ICOUP(39)
      COMMON/POS/ XP(40),YP(40),YAW(40),ROLL(40),XCL(40),YCL(40),
     1 XCT(40),YCT(40)
      COMMON/RUPT/ RUPTYP(40),RUPTD(40),RUPTH(40),RUPTL(40)
C
C     LOCATE CARS
C
      CALL LOCATE(X)
C
C     CHECK FOR RUPTURES
C
      CALL RUPTRE(X)
C
C     GET COUPLER FORCES
C
      CALL CPLFOR
C
C     LOOP OVER ALL THE CARS
C
      ICNT=-6
      DO 10 I=1,NCARS
      ICNT=ICNT+6
C
C     GET APPLIED FORCES
C
      CALL FORCE(I,SFX,SFY,SMG,X)
C
C     ACCOUNT FOR COLLISIONS
C
      IF (I.EQ.1) GO TO 20
      DX=XP(I-1)-XP(I)
      IF (DX.GT.2.0) GO TO 20
      U1=X(ICNT-6+4)
      V1=X(ICNT-6+5)
C     R1=X(ICNT-6+6)
      U2=X(ICNT+4)
      V2=X(ICNT+5)
C     R2=X(ICNT+6)
      ANG1=X(ICNT-6+3)
      ANG2=X(ICNT+3)
      CS1=DCOS(ANG1)
      SN1=DSIN(ANG1)
      CS2=DCOS(ANG2)
      SN2=DSIN(ANG2)
      VX1=U1*CS1-V1*SN1
      IF (VX1.LT.0.) VX1=0.
      VY1=U1*SN1+V1*CS1
      VX2=U2*CS2-V2*SN2
      IF (VX2.LT.0.) VX2=0.
      VY2=U2*SN2+V2*CS2
      VX=0.5*(VX1+VX2)
      X(ICNT-6+4)=VX*CS1+VY1*SN1
      X(ICNT-6+5)=-VX*SN1+VY1*CS1
      X(ICNT+4)=VX*CS2+VY2*SN2
      X(ICNT+5)=-VX*SN2+VY2*CS2
20    CONTINUE
C
C
C     LOAD DERIVATIVE VECTOR
C
      DXDT(ICNT+1)=X(ICNT+4)+X(ICNT+6)*X(ICNT+2)
      DXDT(ICNT+2)=X(ICNT+5)-X(ICNT+6)*X(ICNT+1)
      DXDT(ICNT+3)=X(ICNT+6)
      DXDT(ICNT+4)=SFX/CARM(I)+X(ICNT+5)*X(ICNT+6)
      DXDT(ICNT+5)=SFY/CARM(I)-X(ICNT+4)*X(ICNT+6)
      DXDT(ICNT+6)=SMG/CARI(I)
10    CONTINUE
C
      RETURN
      END
C
C
C
C
      SUBROUTINE FORCE(ICAR,SFX,SFY,SMG,X)
      IMPLICIT REAL*8 (A-H,O-Z)
C
C     CALCULATES APPLIED FORCES
C
      REAL*8 MUG,MUR,X(240)
      INTEGER RUPTYP
      COMMON/PARAM/ CARM(40),CARI(40),ICDRL,NCARS,EL,AD,ANGV,ITYPE(40)
      COMMON/COUPLR/ FCOUP(39),ALPHA(39),ICOUP(39)
      COMMON/POS/ XP(40),YP(40),YAW(40),ROLL(40),XCL(40),YCL(40),
     1 XCT(40),YCT(40)
      COMMON/RUPT/ RUPTYP(40),RUPTD(40),RUPTH(40),RUPTL(40)
      COMMON/INIT/ DRLST,IDRL(40)
      DATA MUR/0.1/,MUG/0.3/,GRAV/9.81/,DWH/2.8E5/,TRAD/2.8/
      DATA IDIR/1/
      IDIR=-IDIR
C
C     CARS ON RAIL AHEAD OF THE DERAILMENT POINT
C
      IF (ICAR.GT.ICDRL-1) GO TO 10
      ICNT=6*(ICAR-1)
      COUPF=-FCOUP(ICAR)
      IF (ICAR.NE.1) COUPF=COUPF+FCOUP(ICAR-1)
      SFX=-MUR*CARM(ICAR)*GRAV*X(ICNT+4)/DABS(X(ICNT+4))+COUPF
      SFY=0.0
      SMG=0.0
      GO TO 77
C
C     CARS WHICH HAVE DERAILED OR ARE NEAR TO THE DERAILMENT
C
10    IF (XP(ICAR).LE.-DRLST) GO TO 20
      IF (IDRL(ICAR).NE.1) IDRL(ICAR)=2
C
C     ADD GROUND FORCES
C
      ICNT=6*(ICAR-1)
C     IF (X(ICNT+6).EQ.0.D0) THEN
C     X(ICNT+5)=10*IDIR*X(ICNT+4)
C     X(ICNT+6)=200.*ANGV*IDIR
C     ENDIF
      U=X(ICNT+4)
      V=X(ICNT+5)
      R=X(ICNT+6)
      VL=V+EL*R
      VT=V-EL*R
      FMAX=0.5*MUG*CARM(ICAR)*GRAV
      IF (DABS(U).LT.0.1 .AND. DABS(V).LT.0.1) FMAX=0.0
      VLM=SQRT(U*U+VL*VL)
      VTM=SQRT(U*U+VT*VT)
      SFX=-FMAX*U/VLM-FMAX*U/VTM
      FYL=-FMAX*VL/VLM
      FYT=-FMAX*VT/VTM
      SFY=FYL+FYT
      SMG=(FYL-FYT)*EL
C
C     CHECK FOR ROLL
C
      IF (ROLL(ICAR).NE.0.D0) GO TO 15
      IF (ITYPE(ICAR).NE.2 .OR. ICAR.EQ.1) GO TO 15
      IF(ICOUP(ICAR-1).NE.0 .OR. ICOUP(ICAR).NE.0) GO TO 15
      IF (SFY.LT.-DWN) ROLL(ICAR)=VL**2/(2.*MUG*GRAV*TRAD)
      IF (SFY.GT.DWN) ROLL(ICAR)=-VL**2/(2.*MUG*GRAV*TRAD)
      IF (ROLL(ICAR).LT.-2) ROLL(ICAR)=-2.
      IF (ROLL(ICAR).GT.2) ROLL(ICAR)=2.
15    CONTINUE
C
C     ADD LEADING COUPLER FORCES
C
      ANGL=ALPHA(ICAR-1)-YAW(I)
      CS=DCOS(ANGL)
      SN=DSIN(ANGL)
      SFX=SFX+FCOUP(ICAR-1)*CS
      FY=FCOUP(ICAR-1)*SN
      SFY=SFY+FY
      SMG=SMG+FY*(AD+EL)
C
C     ADD TRAILING COUPLER FORCES
C
      ANGL=ALPHA(ICAR)-YAW(I)
      CS=DCOS(ANGL)
      SN=DSIN(ANGL)
      SFX=SFX-FCOUP(ICAR)*CS
      FY=FCOUP(ICAR)*SN
      SFY=SFY-FY
      SMG=SMG+FY*(AD+EL)
C
      GO TO 77
C
C     CAR WHICH HAS NOT YET REACHED THE DERAILMENT POINT
C
20    CONTINUE
      COUPF=FCOUP(ICAR-1)
      IF (ICAR.NE.NCARS) COUPF=COUPF-FCOUP(ICAR)
      VCHK=X(6*(ICAR-1)+4)
      SFX=-MUR*CARM(ICAR)*GRAV*VCHK/DABS(VCHK)+COUPF
      SFY=0.0
      SMG=0.0
C
77    RETURN
      END
C
C
C
C
      SUBROUTINE CPLFOR
      IMPLICIT REAL*8 (A-H,O-Z)
C
C     CALCULATES COUPLER FORCES
C
      INTEGER RUPTYP
      COMMON/PARAM/ CARM(40),CARI(40),ICDRL,NCARS,EL,AD,ANGV,ITYPE(40)
      COMMON/COUPLR/ FCOUP(39),ALPHA(39),ICOUP(39)
      COMMON/INIT/ DRLST,IDRL(40)
      COMMON/POS/ XP(40),YP(40),YAW(40),ROLL(40),XCL(40),YCL(40),
     1 XCT(40),YCT(40)
      COMMON/RUPT/ RUPTYP(40),RUPTD(40),RUPTH(40),RUPTL(40)
      DATA CSTF/5.0E7/,BREAK/2.0E7/
C
      DO 10 I=1,NCARS-1
      DX=XCT(I)-XCL(I+1)
      DY=YCT(I)-YCL(I+1)
      ALPHA(I)=0.0
      IF (DX.NE.0.0 .AND. DY.NE.0.0 ) ALPHA(I)=DATAN2(DY,DX)
      DELTA=DX*DCOS(ALPHA(I))+DY*DSIN(ALPHA(I))
      FORC=CSTF*DELTA
      FCOUP(I)=FORC*ICOUP(I)
      IF (IDRL(I).EQ.0) FCOUP(I)=0.
      IF(FCOUP(I).GT.BREAK) ICOUP(I)=0
10    CONTINUE
C
      RETURN
      END
C
C
C
C
      SUBROUTINE LOCATE(X)
      IMPLICIT REAL*8 (A-H,O-Z)
C
C     LOCATES CAR CG'S AND COUPLERS
C
      REAL*8 X(240)
      INTEGER RUPTYP
      COMMON/PARAM/ CARM(40),CARI(40),ICDRL,NCARS,EL,AD,ANGV,ITYPE(40)
      COMMON/COUPLR/ FCOUP(39),ALPHA(39),ICOUP(39)
      COMMON/POS/ XP(40),YP(40),YAW(40),ROLL(40),XCL(40),YCL(40),
     1 XCT(40),YCT(40)
      COMMON/RUPT/ RUPTYP(40),RUPTD(40),RUPTH(40),RUPTL(40)
C
      ICNT=-6
      DO 10 I=1,NCARS
      ICNT=ICNT+6
      YAW(I)=X(ICNT+3)
      CS=DCOS(YAW(I))
      SN=DSIN(YAW(I))
      XP(I)=X(ICNT+1)*CS-X(ICNT+2)*SN
      YP(I)=X(ICNT+1)*SN+X(ICNT+2)*CS
      XCL(I)=XP(I)+(AD+EL)*CS
      YCL(I)=YP(I)+(AD+EL)*SN
      XCT(I)=XP(I)-(AD+EL)*CS
      YCT(I)=YP(I)-(AD+EL)*SN
10    CONTINUE
C
      RETURN
      END
C
C
C
C
      SUBROUTINE RUPTRE(X)
      IMPLICIT REAL*8 (A-H,O-Z)
C
C     CHECKS FOR RUPTURED TANKCARS
C
      REAL*8 X(240)
      INTEGER RUPTYP
      COMMON/PARAM/ CARM(40),CARI(40),ICDRL,NCARS,EL,AD,ANGV,ITYPE(40)
      COMMON/COUPLR/ FCOUP(39),ALPHA(39),ICOUP(39)
      COMMON/INIT/ DRLST,IDRL(40)
      COMMON/POS/ XP(40),YP(40),YAW(40),ROLL(40),XCL(40),YCL(40),
     1 XCT(40),YCT(40)
      COMMON/RUPT/ RUPTYP(40),RUPTD(40),RUPTH(40),RUPTL(40)
      DATA TNKR/4.0D0/
C
      DO 10 I=1,NCARS-1
      IF (ICOUP(I).NE.0) GO TO 10
      XC1=XCT(I)
      YC1=YCT(I)
      DO 20 J=1,NCARS
      D1=DSQRT((XP(J)-XC1)**2+(YP(J)-YC1)**2)
      IF (D1.GT.TNKR) GO TO 20
      IF (IDRL(J).EQ.0) GO TO 20
      ICNT=6*(J-1)
      TESTL=XC1/DCOS(YAW(J))-X(ICNT+1)
      IF (DABS(TESTL).GT.7.0) GO TO 20
      U=X(ICNT+4)
      V=X(ICNT+5)
      R=X(ICNT+6)
      VL=V+TESTL*R
      VMAG=DSQRT(U*U+VL*VL)
      IF (VMAG.LT.5.) GO TO 20
      RUPTYP(J)=2
      IF (U.GT.200.*VL) RUPTYP(J)=1
      RUPTH(J)=0.75
      RUPTL(J)=TESTL
20    CONTINUE
10    CONTINUE
C
      RETURN
      END
C
C
C
C
      SUBROUTINE RKFINT(A,B,Y,DYDX,N,TOL,HSTART,HMIN,HMAX,IFLAG)
C
      IMPLICIT REAL*8 (A-H,O-Z)
      REAL*8 Y(N),DYDX(N),EST(240),YOLD(240)
      COMMON/LOC6/ EST,YOLD
C
      HOLD=HSTART
      X=A
      DO 10 I=1,N
 10   YOLD(I)=Y(I)
 1    CALL RKF(X,Y,DYDX,HOLD,N)
      CALL GAMCAL(A,B,TOL,GAMMA,N)
      HNEW=.8*GAMMA*HOLD
      IF (GAMMA.GE.1.D0) GO TO 2
```

```
C
      IF (HNEW.LT.HOLD/10.D0) HNEW=HOLD/10.D0
      IF (HNEW.LT.HMIN) GO TO 3
      HOLD=HNEW
      GO TO 1
C
 2    IF (HNEW.LT.HMIN) GO TO 3
      IF (HNEW.GT.5.*HOLD) HNEW=5.*HOLD
      IF (HNEW.GT.HMAX) HNEW=HMAX
      IF (X+HOLD.GE.B) GO TO 4
      X=X+HOLD
      HOLD=HNEW
      DO 20 I=1,N
 20   YOLD(I)=Y(I)
      GO TO 1
C
 3    IFLAG=1
      B=X
      DO 30 I=1,N
 30   Y(I)=YOLD(I)
      RETURN
C
 4    IFLAG=0
      HSTART=HNEW
      HSTART=HOLD
      HOLD=B-X
      CALL RKF(X,Y,DYDX,HOLD,N)
C
      RETURN
      END
C
C
C
C
      SUBROUTINE GAMCAL(A,B,TOL,GAMMA,N)
C
      IMPLICIT REAL*8 (A-H,O-Z)
      REAL*8 YOLD(240),EST(240),R(240)
      COMMON/LOC6/ EST,YOLD
      COMMON/LOC62/ R
C
      TOLR=TOL
      TOLA=TOL
      DO 10 I=1,N
      ABSEST=DABS(EST(I))
      IF (ABSEST.EQ.0.D0) GO TO 1
      R(I)=((TOLR*DABS(YOLD(I))+TOLA)/(ABSEST*(B-A)))**.25
      GO TO 10
C
 1    R(I)=6.25D0
C
 10   CONTINUE
```

```
C
      GAMMA=R(1)
      DO 20 I=2,N
20    IF (R(I).LT.GAMMA) GAMMA=R(I)
C
      RETURN
      END
C
C
C
C
      SUBROUTINE RKF(X,Y,DYDX,H,N)
C
C.....RUNGE-KUTTA-FEHLBERG ROUTINE
C
      IMPLICIT REAL*8 (A-H,O-Z)
      REAL*8 K1(240),K2(240),K3(240),K4(240),K5(240),K6(240)
      REAL*8 YOLD(240),Y(N),DYDX(N),EST(240)
C
      COMMON/LOC6/ EST,YOLD
      COMMON/LOC63/ K1,K2,K3,K4,K5,K6
C
      DATA B21/0.250000000000000D+00/
      DATA B31/0.937500000000000D-01/
      DATA B32/0.281250000000000D+00/
      DATA B41/0.879380974055530D+00/
      DATA B42/-0.327719617660446D+01/
      DATA B43/0.332089212562585D+01/
      DATA B51/0.203240740740741D+01/
      DATA B52/-0.800000000000000D+01/
      DATA B53/0.717348927875244D+01/
      DATA B54/-0.205896686159844D+00/
      DATA B61/-0.296296296296296D+00/
      DATA B62/0.200000000000000D+01/
      DATA B63/-0.138167641325536D+01/
      DATA B64/0.452972709551657D+00/
      DATA B65/-0.275000000000000D+00/
      DATA A1/0.115740740740741D+00/
      DATA A3/0.548927875243665D+00/
      DATA A4/0.535331384015595D+00/
      DATA A5/-0.200000000000000D+00/
      DATA R1/0.277777777777778D-02/
      DATA R3/-0.299415204678363D-01/
      DATA R4/-0.291998936735779D-01/
      DATA R5/0.200000000000000D-01/
      DATA R6/0.363636363636364D-01/
C
      DO 10 I=1,N
 10   Y(I)=YOLD(I)
      XT=X
      CALL XDOT(N,XT,Y,DYDX)
      DO 20 I=1,N
      K1(I)=DYDX(I)
20    Y(I)=YOLD(I)+H*B21*K1(I)
      XT=X+H/4.D0
      CALL XDOT(N,XT,Y,DYDX)
      DO 30 I=1,N
      K2(I)=DYDX(I)
30    Y(I)=YOLD(I)+H*(B31*K1(I)+B32*K2(I))
      XT=X+3.D0*H/8.D0
      CALL XDOT(N,XT,Y,DYDX)
      DO 40 I=1,N
      K3(I)=DYDX(I)
40    Y(I)=YOLD(I)+H*(B41*K1(I)+B42*K2(I)+B43*K3(I))
      XT=X+12.D0*H/13.D0
      CALL XDOT(N,XT,Y,DYDX)
      DO 50 I=1,N
      K4(I)=DYDX(I)
50    Y(I)=YOLD(I)+H*(B51*K1(I)+B52*K2(I)+B53*K3(I)+B54*K4(I))
      XT=X+H
      CALL XDOT(N,XT,Y,DYDX)
      DO 60 I=1,N
      K5(I)=DYDX(I)
60    Y(I)=YOLD(I)+H*(B61*K1(I)+B62*K2(I)+B63*K3(I)+B64*K4(I)
     1  +B65*K5(I))
      XT=X+H/2.D0
      CALL XDOT(N,XT,Y,DYDX)
      DO 70 I=1,N
      K6(I)=DYDX(I)
      Y(I)=YOLD(I)+H*(A1*K1(I)+A3*K3(I)+A4*K4(I)+A5*K5(I))
70    EST(I)=R1*K1(I)+R3*K3(I)+R4*K4(I)+R5*K5(I)+R6*K6(I)
C
      RETURN
      END
```

6. REFERENCES

1. Yang, T.H., Manos, W.P. and Johnstone, B., Dynamic Analysis of Train Derailments, ASME 72-WA/RT-6, ASME Winter Annual Meeting, November, 1972.

2. Yang, T.H. et al., A Study Continuation of Derailment Behavior: Final Report, Pullman-Standard Research Project No. 39-1833, RPI-AAR Railroad Tank Car Safety Research and Test Project, Report RA-08-1-12, February, 1972.

3. Phillips, E.A. and Role, H., Effectiveness of Shelf Couplers, Head Shields and Thermal Shields on DOT 112(114) and 105 Tank Cars, RPI-AAR Railroad Tank Car Safety Research and Test Project, Report RA-02-5-51, June, 1985.

4. Ward, E.D. and Leonard, R.G., Automatic Parameter Identification Applied to a Railroad Car Dynamic Draft Gear Model, ASME 74-WA/Aut-1, ASME Winter Annual Meeting, November, 1974.

5. Wallace, W.D., Evaluation of Railway Draft Gears, ASME Paper Number 57-RR-8, Anthology of Rail Vehicle Dynamics, Vol.I, Freight Car Impact, pp.1-8, 1971.

6. Raidt, J.B., Manos, W.P. and Johnstone, B., Vertical Motions During Railcar Impacts, ASME 75-WA/RT-10, ASME Winter Annual Meeting, 1975.

7. Wikander, O.R., Draft-Gear Action in Train Service, ASME Paper Number 44-SA-13, Anthology of Rail Vehicle Dynamics, Vol.II, Effects ot Train Action and Rail Car Vibration, pp.3-9, 1972.

APPENDIX D.2

SUBROUTINE BLAST

APPENDIX D.2

SUBROUTINE BLAST

Subroutine BLAST is used to calculate the blast effects of an unconfined vapour cloud explosion.

The initial mass of the explosive is assumed to equal the total mass of lading in the tank. This mass is then converted to an equivalent mass of TNT using data from Reference (1). Table D.2-1 contains a partial listing from the above reference.

Table D.2-1 Conversion Factors for TNT Equivalency (From Ref. (1))

COMPOUND (1 kg)	EQUIVALENT TNT MASS (kg)
methane	11.95
ethane	11.34
propane	11.07
butane	10.91
ethylene	11.26
propylene	10.94

The equivalent TNT mass is then reduced by multiplication by an explosion efficiency for unconfined vapour cloud explosions. As discussed in Ref. (2), this explosion efficiency varies widely for unconfined vapour cloud explosions. As shown in Figure D.2-2, the range of efficiencies is between .05 and .65. The efficiency of the blast is determined by the vapour cloud conditions at the time of ignition. As discussed in Reference (1), the strength of the blast is dependent on whether there is a detonation or a deflagration. If the cloud has mixed with the surrounding air, then a detonation is likely. If, however, poor mixing has taken place, then a deflagration is likely.

---

(1)

(2)

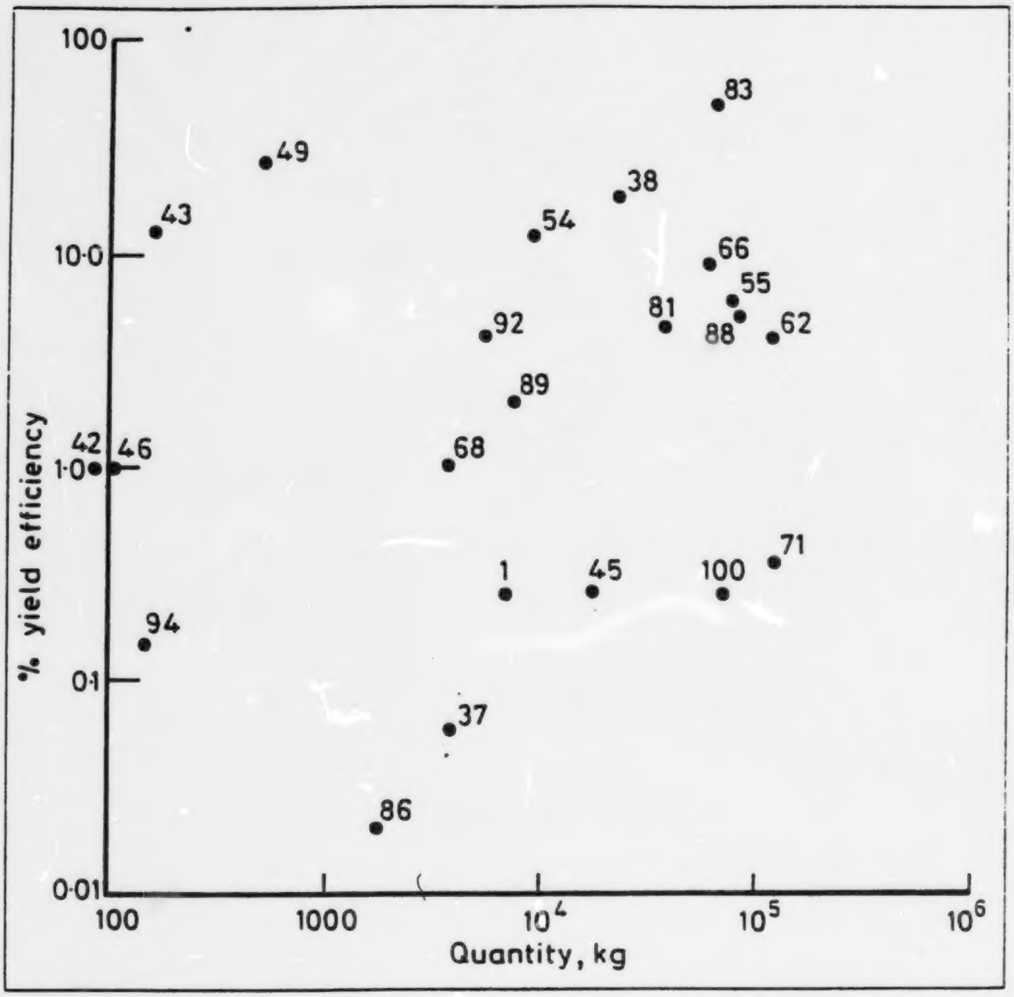


Figure D.2-2 Assessed Efficiency of Unconfined Vapour Cloud Explosions

In BLAST, it is assumed that if a BLEVE occurs, the explosive efficiency wil be high (i.e. = .65) while if a BLEVE is unlikely, the efficiency will be low (i.e. = .10).

Based on the TNT equivalent mass and the explosive efficiency, it is possible to estimate the free field overpressure as a function of distance using the following emplirical relation.

$$Po/Pa = \frac{808\ (1 + (z/\ 4,5\ )^2}{(1+(z/0.048)^2)^{\frac{1}{2}}\ (1+z/0.32)^2)^{\frac{1}{2}}\ (1+z/1.35)^2)^{\frac{1}{2}}}$$

where P^o = overpressure

P_a = atmospheric pressure

z = scaled distance (m)

= $\left(\frac{Pa}{P_o}\right)^{.333} \left(\frac{To}{T}\right)^{.333} \frac{d}{W^{.333}}$

d = distance from centre of explosion (m)

P_ao = reference atmospheric pressure (kPa)

T_a = ambient air temperature (K)

T_ao = reference ambient air temperature (K)

W = equivalent mass of TNT (kg)

Subroutine BLAST then uses three preselected free-field overpressures to calculate hazard distances from the centre of explosion. Overpressure levels and their associated damage estimates are presented in Ref. (1) and summarized in Table D.2-3.

Table D.2-3 Blast Damage - Side on Overpressure Correlation (Large Explosion)

Type of Damage	Millibars
Minimum damage to glass panels	1-3
Typical window glass breakage	10-15
Overpressure at limit for debirs and missile damage	15-25
Windows shattered, plaster cracked; minor damage to some building	35-75
Personnel knocked down	70-100
Panels of sheet metal buckled	75-125
Failure of wooden or asbestos sliding for conventional homes	75-150
Failure of walls constructured of concrete blocks or cinder blocks	125-200
Self-framing paneled buildings collapse	200-300
Oil storage tanks ruptured	200-300
Utility poles broken off	300-500
Serious damage to buildings with structural steel framework	300-500
Eardrum rupture	350-1000
Reinforced concrete structures severely damaged	400-600
Railroad cars overturned	400-600
Probable total destruction of most buildings	700-800
Lung damage	2000-5000
Lethality	7000-15000
Crater formation in average soil	20000-30000

In BLAST, the three levels of damage selected are:

Damage	Overpressure	
Lethal	7000	m bar
Destruction of buildings	700	
Storage tank ruptured	200	

For a given explosion of a tank's contents, BLAST determines the distances to the above levels of damage. These are then displayed by the DERACS post processor.

References

(1) Kinney, G.F., Graham, K.J., Explosive Shocks in Air, 2nd Ed., Springer-Verlag, 1985.

(2) Gugan, Keith, Unconfined Vapor Cloud Explosions, The Institute of Chemical Engineers, 1979.

APPENDIX D.3

SUBROUTINE BLEVE

APPENDIX D.3

SUBROUTINE BLEVE

This subroutine is used to determine if a BLEVE (boiling liquid expanding vapour explosion) is likely in the event of a total loss of containment of a tank-car carrying propane.

The calculation is based on the theory for explaining BLEVEs from Reference (1). In this theory, it is concluded that for a BLEVE to take place, the entire liquid mass must vaporize instantaneously. For this to occur, the entire liquid mass must cross the super heat limit locus.

When a pressurized liquid sees a reduction in pressure, it will boil if bubble nucleation sites are present and vapour will be released. Heat is required to convert liquid to vapour and this heat comes from the lading itself and other sources. If the heat comes from the liquid, its temperature will drop until an equilibrium is reached and the liquie will stop boiling. This is the saturated equilibrium process.

However, if the pressure drop is very rapid, then the liquid will superheat before it boils. The superheat limit locus (shown in Figure D.3-1) defines the limit to which the liquid can superheat befgore homogenous nucleation occurs. If the entire liquid mass reaches the superheat limit locus and then nucleates homogeneously, a vapour explosion (BLEVE) can result.

As shown in Figure D.3-1, propane will not nucleate homogenously at 1 atm pressure below 53°C. That is, starting from an initial condition at A, if the pressure is suddenly reduced to 1 atm, a vapour explosion will not result. However, starting at position B, if the pressure is reduced to 1 atm suddenly, then homogenous nucleation will begin at the superheat limit locus pressure at C.

In subroutine BLEVE, a simple check is performed. If the lading saturation temperature is above 53°C, then a total loss of containment will be assumed to result in a BLEVE. If, on the other hand, the lading saturation temperature is below 53°C, then it is assumed that a BLEVE will not occur.

---

(1)

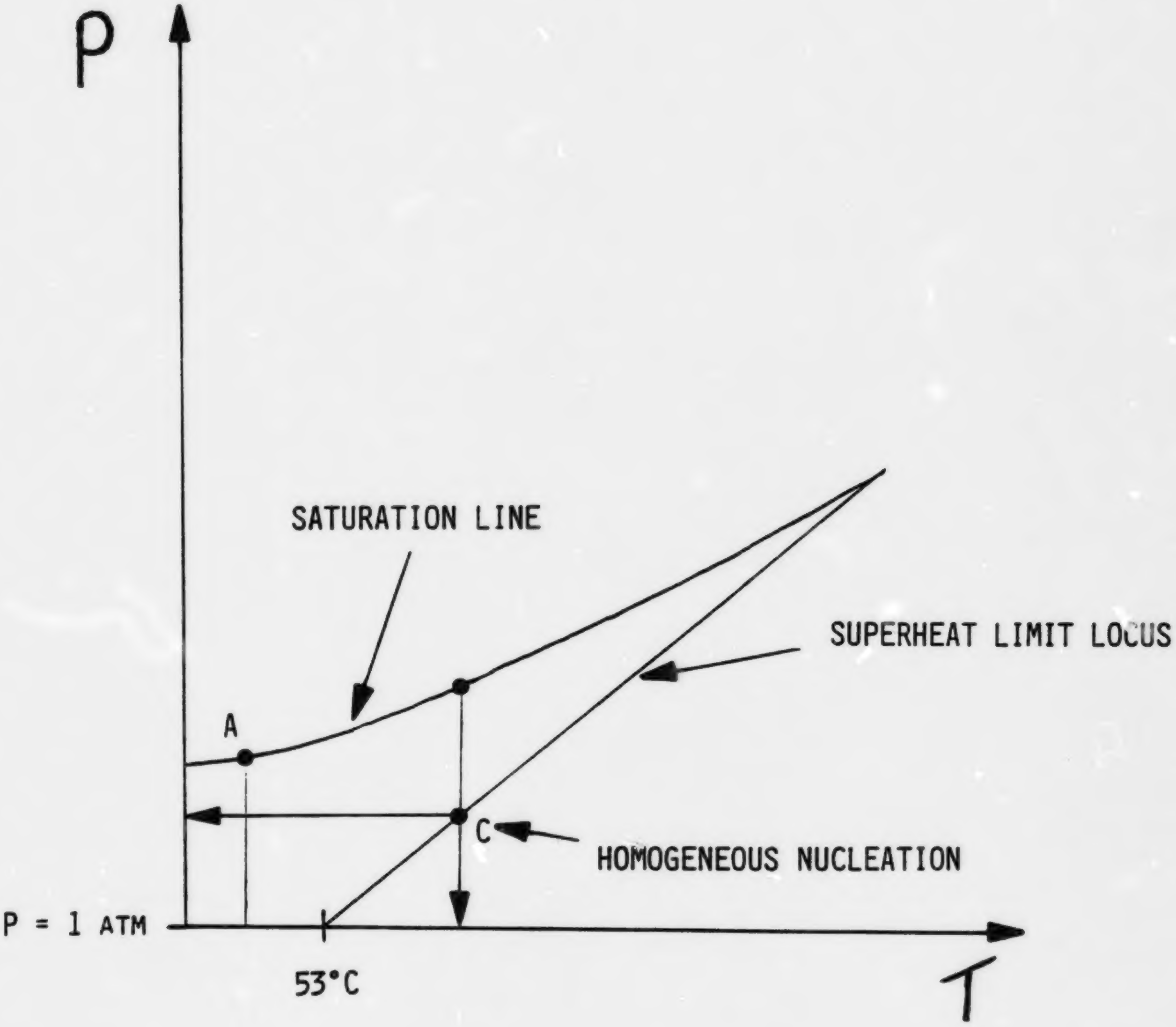

P
SATURATION LINE
SUPERHEAT LIMIT LOCUS
A
C
HOMOGENEOUS NUCLEATION
P = 1 ATM
53°C
T

Figure D.2-1 Superheat Limit Locus for Propane

References

(1) Reed, R.C., Possible Mechanism for Pressurized Liquid Tank Explosions or BLEVEs, Science, Vol. 203, March 1979.

APPENDIX D.4

SUBROUTINE CONREL

SUBROUTINE CONREL

As stated in the body of this report, CONREL is a modified version of the LEAKR program of Reference (1).

LEAKR is a standalone program for predicting releases for punctured tanks. The program can account for a variety of puncture types and sizes, tank shapes, commodity types, etc. The program predicts the outflow conditions as a function of time and predicts the total time required to empty the tank.

For use in DERACS, the LEAKR program was modified in the following areas:

i) The time integration function of the program was removed. This was done because DERACS carries out the time integration and CONREL was needed only to calculate an instantaneous flow rate given the current conditions in the tank at a given time step.

ii) LEAKR calculates the regression of the liquid surface in constant increments of 5 cm. This was changed from a dimension step to a time step of 20 seconds. Therefore, when CONREL is called, the current mass flow rate is calculated and the lading mass is decreased by the amount of lading removed in 1 time step. In other words, LEAKR calculates mass flow over varying time steps while CONREL uses a constant time step of 20 seconds.

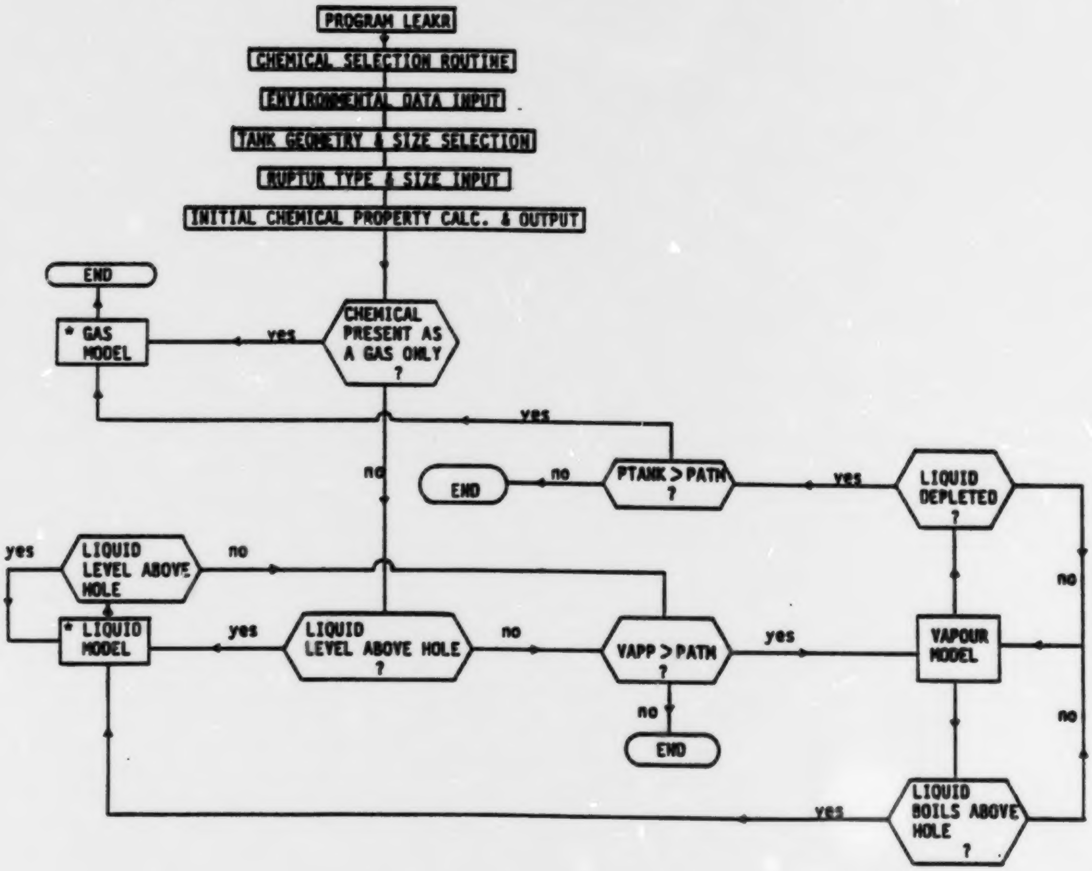
iii) The calculation procedure was simplified in LEAKR to account for the fact that for large punctures, the liquid and vapor can be expelled in less than one time step.

The LEAKR program is complex and every attempt was made not to invalidate its calculation procedure. Although limited testing and verification was performed, it is possible that errors have been made and therefore care must be taken when interpreting results generated by CONREL. Future work on DERACS should include a proper evaluation of CONREL.

The reader is directed to the source listing of CONREL in Appendix B for details. Figure D.4-1 shows the main program structure of LEAKR from Ref. (1). Figure D.4-2 shows approximately the main program structure of CONREL.

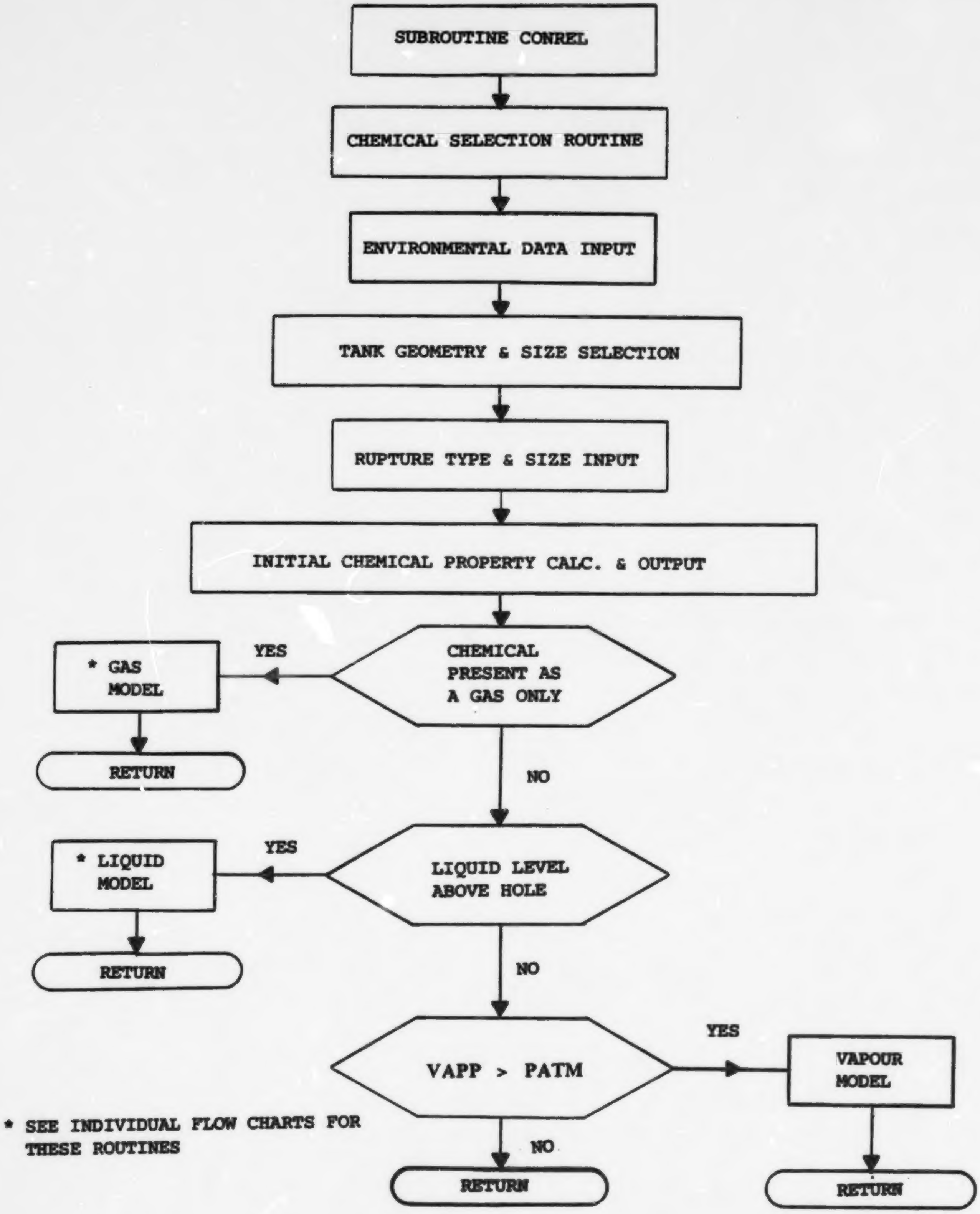
---

(1)


PROGRAM LEAKR
CHEMICAL SELECTION ROUTINE
ENVIRONMENTAL DATA INPUT
TANK GEOMETRY & SIZE SELECTION
RUPTUR TYPE & SIZE INPUT
INITIAL CHEMICAL PROPERTY CALC. & OUTPUT
END
* GAS MODEL
yes
CHEMICAL PRESENT AS A GAS ONLY ?
yes
no
END
no
PTANK > PATH ?
yes
LIQUID DEPLETED ?
yes
LIQUID LEVEL ABOVE HOLE
no
no
* LIQUID MODEL
yes
LIQUID LEVEL ABOVE HOLE ?
no
VAPP > PATH ?
yes
VAPOUR MODEL
no
END
no
yes
LIQUID BOILS ABOVE HOLE ?

* SEE INDIVIDUAL FLOW CHARTS FOR
THESE ROUTINES

Figure D.4-1 Mainline Program Structure for LEAKR


SUBROUTINE CONREL
CHEMICAL SELECTION ROUTINE
ENVIRONMENTAL DATA INPUT
TANK GEOMETRY & SIZE SELECTION
RUPTURE TYPE & SIZE INPUT
INITIAL CHEMICAL PROPERTY CALC. & OUTPUT
CHEMICAL PRESENT AS A GAS ONLY
YES
* GAS MODEL
RETURN
NO
LIQUID LEVEL ABOVE HOLE
YES
* LIQUID MODEL
RETURN
NO
VAPP > PATM
YES
VAPOUR MODEL
NO
RETURN
RETURN

* SEE INDIVIDUAL FLOW CHARTS FOR THESE ROUTINES

Figure D.4-2 Subroutine Structure for CONREL

References

1) Belore, R., Buist, I., A Computer Model for Predicting Leak Rates of Chemicals from Damaged Storage and Transportation Tanks, Report EE-75, Environmental Emergencies Technology Division, Environment Canada.

APPENDIX D.5

SUBROUTINE FIRBAL

APPENDIX D.5

SUBROUTINE FIRBAL

Subroutine FIRBAL is used to calculate the thermal radiation resulting from a fireball. The fireball is assumed to occur if a tank containing propane is thermally ruptured resulting in a total loss of containment.

As discussed in Ref. (1), the consequences of a total loss of containment of a pressurized flammable liquid depends on the conditions at release. The consequences range from a prolonged burnoff to a major fireball and/or vapour cloud explosion.

For the case of the propane car exposed to external fire impingement, the thermal induced rupture will usually result in a total loss of containment. In the case of total loss of containment, part of the lading is rapidly vapourized due to flashing and large quantities of liquid are entrained in the form of liquid droplets. In the case of BLEVE, the flashing is so rapid that an explosion results (hence boiling liquid expanding vapour explosion). The resulting cloud of flammable liquid usually results in a fireball.

It is difficult to estimate how much of the total lading is involved with the fireball. Reference (1) states that if more than 35% of the liquid flashes during the loss of containment, then all of its fuel will be consumed in the fireball. If less than 35% flashes, then some liquid fuel will remain to burn off slowly. In FIRBAL it is assumed that all of the lading is consumed in the fireball.

An empirical relation reported in Reference (2) gives the maximum fireball diameter as being:

$D_{max} = 5.33\ m^{.327}$

where m = fuel mass (kg)

D_{max} = max. fireball diameter (m)

This relation is based on experiments for fuel masses between 3 - 31 kg; however, the validity of the above equation has been verified for much larger masses (2).

The fireball effective duration as described in (2) can be predicted using the following empirical relation:

$t_a = .923\ m^{.303}$

where m = fuel mass (kg)

t_a = time (seconds)

---

(1)

(2)

During the development of a fireball, the diameter grows and, due to buoyancy, the fireball rises off the ground. For the purposes of calculating thermal radiation levels, it has been assumed in FIRBAL that the fireball has a diameter $D_m$ax for time t_a and it is positioned with its bottom edge at ground level. The emissive power of the fireball is assumed to be (from (2)):

$$E = E_{max} (1 - 3^{-bD})$$

where

E_{max} = 469 kW/m²

b = .18 1/m

D = fireball diameter

With the emissive power of the fireball established, it is then a geometry problem to establish the flux reaching a remote target.

Subroutine FIRBAL is used to calculate the distance from the point on the ground centered under the pool fire to a position that receives a certain level of flux. Reference (3) shows dose levels that result in various types of damage. Table D.5-1 presents a summary of these dosages.

Table D.5-1 Dose Criteria

Dose kJ/m²	Effect
37800	heavy equipment damage
12600	building fires
375	3rd degree burns
250	2nd degree burns
125	1st degree burns
65	pain

from Reference (3).

---

(3)

The calculated dose is based on the following relation:

$$Q_D = F\,E\,t_a\,\tau$$

where F = radiant view factor between fireball and target

E = emissive power of fireball (kW/m^2)

t_a = effective fireball duration

τ = atmospheric transmissivity of the fireball radiation

Q_D = dose kJ/m^2

The atmospheric transmissivity is dependent on the path length through the atmosphere, the radiating characteristics of the fireball, and the absorbing characteristics of the atmosphere. An average value of τ is calculated in FIRBAL based on the following equation given in Reference (4).

$$\tau = e^{(-7 \times 10^{-4}L)}$$

where L = path length from fireball edge to target (m).

The radiant exchange factor F is based on the assumption that the radiation can be modelled as that from a sphere of diameter D to a target oriented towards the sphere at a distance R from the point on the ground directly below the fireball.

From Reference (4), the view factor is:

$$F = \frac{(H/D + .5)}{4\,\{(H/D + .5)^2 + (R/D)^2\}^{3/2}}$$

where H is the ground clerance of the fireball, here assumed to be $H = 0$.

Subroutine FIRBAL calculates the distances from a predicted fireball to the three levels of raditaion as indicated below:

distance 1 → 3rd degree burns
2 → 2nd degree burns
3 → 1st degree burns

These distances are then displayed for an explosion scene using the DERACS postprocessor.

---

(3)

(4)

References

(1) Roberts, A.F., The Effect of Conditions Prior to Loss of Containment on Fireball Behavior, F Chem E Symposium, Series No. 71.

(2) Moorehouse, J., Pritchard, M.J., Thermal Radiation from Large Pool Fires and Fireballs: A Literature Review, I Chem E., Symposium, Series No. 71

(3) Crocker, W.P., Napier, D.H., Qualification of Fire Hazards of Liquid Spills from Tank Trucks, Procedures of the Technical Seminar on Chemical Spills, Environment Canada, Feb. 1987.

(4) Litiou, D.A., Maund, J.K., Thermal Radiation Hazard from Fireballs, I Chem E. Symposium, Series No. 7.

APPENDIX D.6

SUBROUTINE FIRIMP

APPENDIX D.6

SUBROUTINE FIRIMP

This subroutine is called to establish whether the spill associated with the current tank-car (ITCAR) impingines on any of the other tank-cars in the simulations.

Consider the case shown in Figure D.6-1, which shows a ruptured tank-car sitting in a pool of liquid. The pool would be hexane in the present version of DERACS. Car #1 is the source of the hexane that formed the pool and car #2 is one other tank-car in the simulation.

The jobto be performed is to determine whether car #2 is impinged by the pool fire created by the rupture of car #1.

The first calculation involves the translation from the x^1, y^1 coordinate system to the x, y coordinate system located relative to the center of the pool. The x, y coordinates of car #2 relative to this system are:

$$x_2 = x'_2 - x'_1$$

$$y_2 = y'_2 - y'_1$$

now the end point of the tank are located as

$$x_A = x + \frac{L}{2} \cos \theta$$

$$x_B = x - \frac{L}{2} \cos \theta$$

$$y_A = y - \frac{L}{2} \sin \theta$$

$$y_B = y - \frac{L}{2} \sin \theta$$

The next calculation determines the distance from the pool center to the tank center and ends.

$$R_o = \sqrt{x^2 + y^2}$$

$$R_A = \sqrt{x_A{}^2 + y_A{}^2}$$

$$R_B = \sqrt{x_B{}^2 + y_B{}^2}$$

If all of the above distances R_o, R_A and R_B are larger than the pool radius, R_p, then the tank is not impinged by the pool fire and the calculation ends.

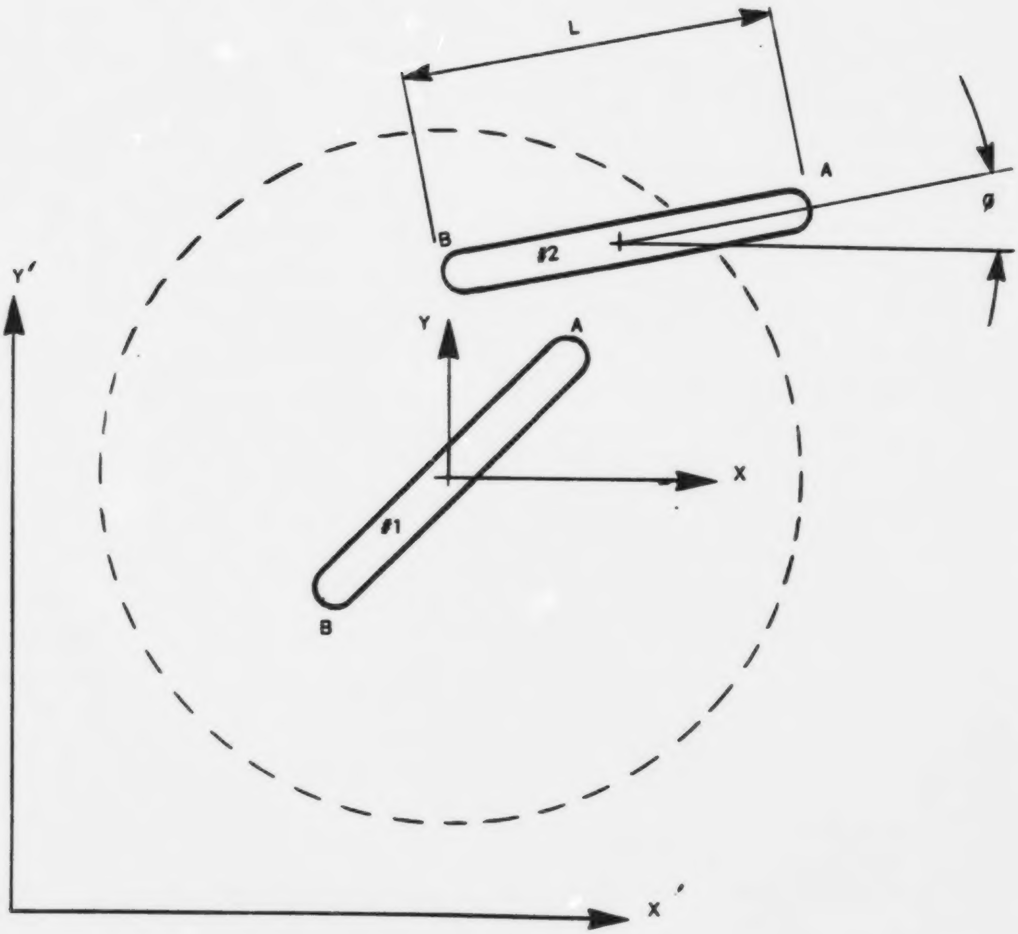

L
A
B
#2
Ø
Y'
Y
A
X
#1
B
X'

FIGURE D.6-1 SAMPLE SCENE OF TANKS IMPINGED BY POOL FIRE

On the other hand, if R_A and $R2_B$ are both smaller than R_p, then the tank is engulfed by the fire.

For the case of partial engulfment, it is necessary to calculate the percentage of the tank impinged by the fire. The calculation begins by determining the equation of the line running through the tank axis.

$$y = \frac{y_B - y_A}{x_B - y_A} \; x \; \frac{0 - x_B}{y_A - x_B} (y_A - y_B) + y_B$$

$$= m_x + b$$

the intersection of this line with the pool edge is when

$$x = \pm \sqrt{R_p^2 - y^2}$$

or,

$$x = \frac{-mb \pm \sqrt{R_p^2 + m^2 R_p^2 - b^2}}{1 + m^2}$$

and the y coordinates of the two intersection points are determined from the equation of the line through the tank.

Figure D.6-2 shows the points that have been identified. With these, it is possible to calculate the percentage of tank engulfment.

The following logic is used to calculate the ratio of the total length of the tank in the fire.

first, let $$l = \frac{x_A - x_B}{2}$$

= ½ the length of the tank projected along the x axis.

If the tank center is located to the right of x_n, then less than half of the tank is in the fire, and:

$$\text{Ratio} = \frac{1 - (x_O - x_N)}{2l}$$

where ratio is the fraction of the tank in the fire.

If the center of the tank is to the left of x_m, then again less than half the tank is in the fire and

$$\text{Ratio} = \frac{1 - (x_m - x_O)}{2l}$$

If all of the tank is located to the left of x_m, then

$$\text{Ratio} = \frac{1 - (x_m - x_B)}{2l}$$

If all of the tank is located to the right of x_m, then

$$\text{Ratio} = \frac{1 - (x_m - x_B)}{2l}$$

and, finally, if both ends of the tank are free of the fire but the middle of the tank is in the fire, then

$$\text{Ratio} = \frac{x_N - x_M}{2l}$$

From the above calculations, the fraction of the tank in the fire has been predicted. The final function of FIRIMP is to set the constants that are stored in the fire impingement matrices IFIREM (I, J) and FDATAM (I, J) where

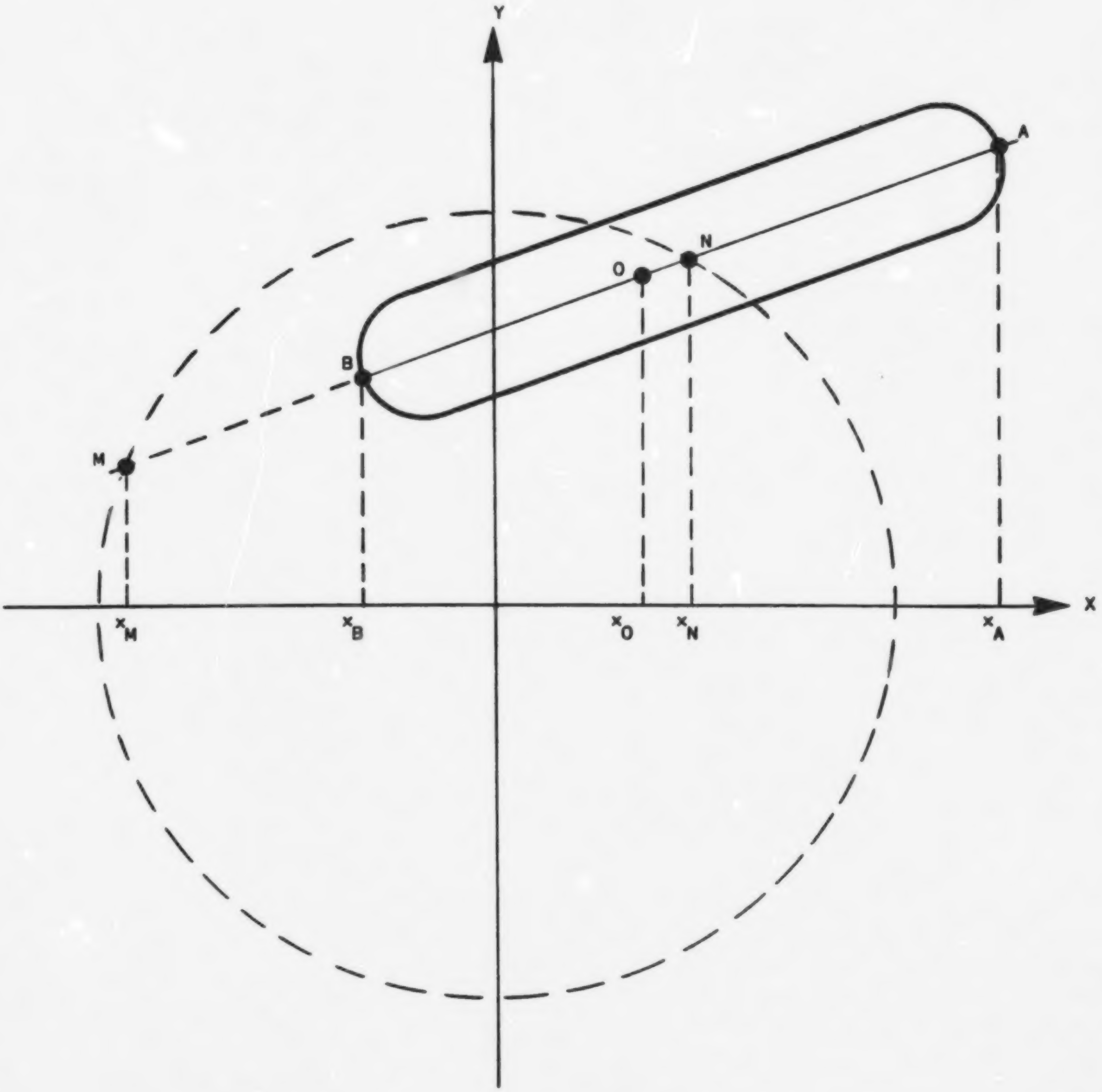
IFIREM (I, J) = 0 no fire
= 1 engulfment
= 2 torch
= 3 partial engulfment

FDATAM (I, J) = 0 no fire
= 1 engulfment
= distance from torch source for torch
= Ratio for partial engulfment

and

I = current car
J = other car causing fire impingement

These two matrices will be used in SETFIR to establish the fire impingement conditions to be used by the tank-car thermal model.



Y
X
A
N
O
B
M
x_M
x_B
x_O
x_N
x_A

FIGURE D.6-2 SKETCH SHOWING POINTS USED TO ESTABLISH PERCENTAGE OF TANK IN FIRE

APPENDIX D.7

SUBROUTINE FLARE

APPENDIX D.7

SUBROUTINE FLARE

This subroutine is used to calculate the length of a burning flare for the case of a tank rolled to near 90° from the vertical ($80° \rightarrow \eta \rightarrow 100°$). This flare length is used by other routines to establish whether other tanks are impinged by the flare. The flare length is also displayed by the post processor.

Figure D.7-1 shows the appearance of a burning flare on a tank of various roll angles. The important feature to note is that the trajectory of the flare, and hence its effective length, is significantly effected by flame buoyancy.

The degree to which buoyancy alters the flame trajectory is a function of the flame Richardson number Ri_s where, as described in Reference (1), this is defined as:

$$Ris = \pi g P\propto D_S^3/4\dot{G}_o$$

where

g = acceleration due to gravity (m/s^2)

$P\propto$ = density of ambient air (kg/m^3)

D_s = effective source diameter

$= \sqrt{4G_o / \pi P\propto U^2_s}$

U_s = source momentum mean velocity (m/s)

$\dot{G}_o$ = source momentum (N)

The vertical flare length in a tank with no roll can be estimated using flame length correlations for the turbulent diffusion of flames. From Reference (2), the flame length can be calculated

$$\frac{L}{D_s} = 63.4\ Ri_s^{-.130}$$

for $10^{-5} < Ris < .005$

The above is based on experimental data. Reference (3) studied the effect of flame tilt on the effective length of the flame. Based on the results presented in Reference (3), the following approximate relation was developed for the flame equivalent length along the ground:

for $\xi \leq 3$ $\quad \frac{x}{L} = 1$

for $\xi > 3$ $\quad \frac{x}{L} = 1 - .087\ (\xi - 3)$

where,

$$\S = \mathrm{Ris}^{.333} \frac{L}{D_s}$$

x = equivalent flare length along the ground (as shown in Figure D.7-1) (m)

The effective flare length along the ground is needed to predict flare impingement on other tanks. It should be noted that the analysis just presented does not account for the effects of wind speed or direction on the flare trajectory. This, of course, is a very important consideration and should be included in later version of this subroutine.

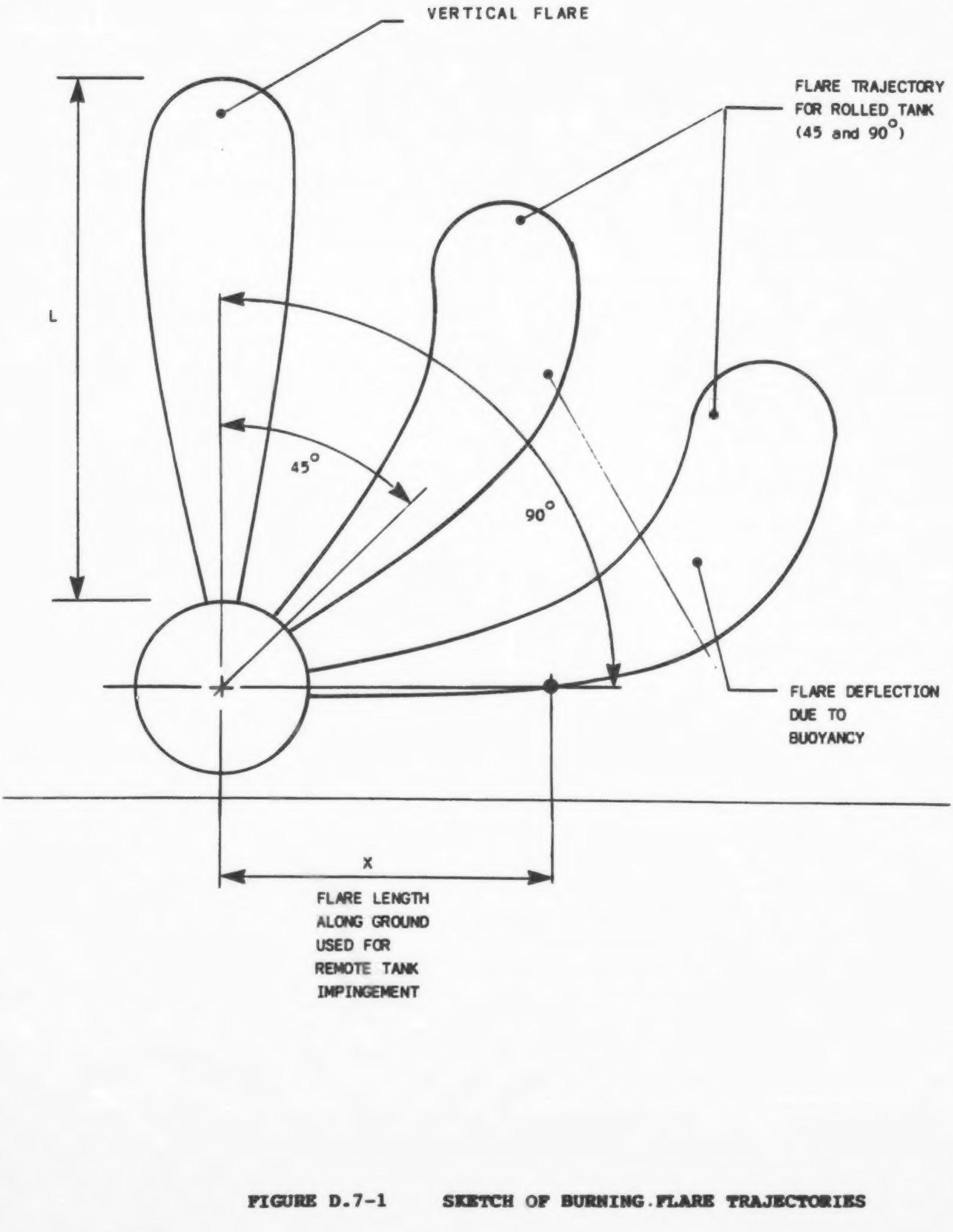

VERTICAL FLARE
FLARE TRAJECTORY FOR ROLLED TANK (45 and 90°)
L
45°
90°
FLARE DEFLECTION DUE TO BUOYANCY
X
FLARE LENGTH ALONG GROUND USED FOR REMOTE TANK IMPINGEMENT

FIGURE D.7-1 SKETCH OF BURNING FLARE TRAJECTORIES

References

(1) Becker, H.A., Liang, D., Total Emission of Soot and Thermal Radiation by Free Turbulent Diffusion Flames, Combusion and Flame 44: 305-318, 1982.

(2) Becker, H.A., Liang, D., Visible Length of Vertical Fire Turbulent Diffusion Flames, Combusion and Flame 32, 115-137, 1978.

(3) Becker, H.A., Liang, D., Downey, C.I., Effect of Burner Orientation and Ambient Airflow on Geomtry of Turbulent Free Diffusion Flames, Eighteenth International Symposium on Combustion, 1981.

APPENDIX D.8

SUBROUTINE PFIRE

APPENDIX D.8

SUBROUTINE PFIRE

Subroutine PFIRE is used to calculate the thermal radiation effects of pool fires on remote targets such as buildings, equipment or people.

The thermal radiation from the pool fire is calculated by the method described in Reference (1). The analysis calculates the radiant heat transfer assuming the pool fire can be treated as a right circular cylinder with the diameter equal to the pool diameter and the height based on experimental data for large pool fires. Reference (2) shows that large hydrocarbon fires have height to diameter ratios of between 1.5 and 2.0. For the present case, it has been assumed that the fire (modelled as a cylinder) height is 1.7 times its diameter.

The heat flux from the fire to a remote target can be calculated using the following selection,

$$q = F\,\Upsilon\,E$$

where,

q = incident flux at target kW/m^2

F = view factor from fire to target

E = emissive power of fire (kW/m^2)

Υ = atmospheric transmissivity

The flame to target view factor F was calculated using the follwing formula based on one given in Reference (3) for a cylinder radiativity to a remote target, forcing torwards the cylinder.

$$F = 1/S(1-1/\Pi(U-(((H^2+S^2+1)H.V)/B^2+4H^2)^{\frac{1}{2}})+H.W)$$

where,

$U = \arccos(H^2-S^2+1)/(H^2+S^2-1)$

$W = \arccos(1/S)$

$V = \arccos(H^2-S^2+1)/((H^2+S^2-1)S)$

$S = x/R$

$H = L/R$

x = distance from center of fire cylinder along the ground (m)

R = pool radius (m)

The atmospheric transmissivity is an approximate value given by the follwoing relationship from Reference (4).

$$\tau = e^{-7.0x10^{-4}x}$$

where x was defined above as the distance from the fire center.

The fire emissive power is a functin of the fire temperature, and the radiating characteristics of the combustion products. Large pool fires tend to generate large amounts of soot which increases the amount of radiated energy. The emissive power for pool fires is calculated in PFIRE using the following relation from Reference (5).

$$E = 250.\ (1 - e^{-.2Dp})$$

where D_p = pool diameter (m)

This expression shows that as the pool fire size increases, the emissive power reaches a limiting flux of 250 kW/m². This limiting value results from the flame acting as a black body radiating at approximately 990°C.

The resulting flux to the target is a function of the fire size and the distance from the fire. In PFIRE, distances are calculated to three preselected levels of flux which represent danger to property or people.

The damage or danger is a result of a product of flux level and duration. The following flux levels will result in the stated damage after some period of time.

human pain (after ≃ 15 seconds)	4.7 kW/m²
wood building ignition	12.6 kW/m²
damage to other vessels	37.8 kW/m²

These three levels of flux are used to calculate three danger distances that are displayed using the DERACS post processor.

References

(1) Crocker, W.P., Napier, D.H., Qualification of Fire Hazards of Liquid Spills from Tank Trucks, Proceedings of the 4th Technical Seminar on Chemical Spills, 1987, Environment Canada, EN 40-327/1-1987.

(2) Modak, A.T., Thermal Radiation from Pool Fires, Combustion and Flame 29, 177-192, 1977.

(3) Howell, J.R., A Catalog of Radiation Configuration Factors, McGraw Hill Book Company

(4) Litiou, D.A., Maund, J.K., Thermal Radiation Hazard from Fireballs, I Chem E. Symposium, Series No. 7, Pergamon Press, Oxford, 1982.

(5) Moorehouse, J., Pritchard, M.J., Thermal Radiation from Large Pool Fires and Fireballs. A Literature Review, I Chem E., Symposium, Series No. 71, Pergamon Press, Oxford, 1982.

APPENDIX D.9

SUBROUTINE SPILL

APPENDIX D.9

SUBROUTINE SPILL

Subroutine SPILL is used to predict the diameter of pools that are created by liquid spills. In DERACS, the only commodity that will remain liquid at atmospheric conditions is Hexane. This commodity then will be the source of pool fires if tanks containing it are ruptured. If a spill of Hexane occurs, it will be assumed that an ignition source is present and therefore a pool fire will result.

The size of the pool is important in determining if other tanks are impinged by the pool fire, and also, pool diameter will affect the fire duration.

Several models exist for predicting the size of pools resulting from liquid spills. The more complex account for ground roughness, permiability, and eleveation. Such complex models are beyond the sope of this prototype version of DERACS which does not include any of the above in sufficient detail for the purposes of these models.

Other simple models are available as described in Reference (1). In this reference, two different spill models are described, one for instanteous spills, as might result from a total loss of containment of a tank containing Hexane, and the other for time dependent spills which would result from a finite puncture below the liquid level in a tank containing Hexane.

In the instantaneous spill model described in Reference (1), it is assumed that the spill surface is flat, smooth and impermiable. This model was coded and it was found taht it predicted very large spill diameters, and very short burn times. The burn times were calculated assuming a constant surface regression rate z (m/s) (the rate at which the pool surface drops due to liquid evaporating and burning). The constant regression rate was taken to be 0.000083 m/s from Reference (2). Such short times will not likely result in tank thermal ruptures (based on the full scale fire tests of Reference (3), an uninsulated tank-car containing propane took 24 minutes to rupture when engulfed in fire). Tanks do rupture due to fire impingement and therefore for the DERACS simulation it was necessary to find a suitable spill model that would yield sufficiently long fire durations.

The second spill model described in Reference (1) requires the specification of a uniform pool depth during the pool spread. This model is based on mass conservation where the spill rate (mass flow rate of the spilling liquid) is equal to the mass of fuel in the spill less the fuel burned off and absorbed by the ground. The problem is to specify a spill depth.

For the purposes of this prototype version of DERACS, a spill model was developed by assuming that the ground at the location of the spill formed a bowl of radius R. As liquid is spilled, it collects in the bowl and based on the mass conservation statement made above, it is possible to estimate the mass of fuel in the bowl. In the present model, it has been assumed that the ground is impermiable and the radius of curvature is $R = 1500$ m.

The radius of the spill can then be calculated based on simple geometry and the fuel density.

Figure D.9-1 presents the geometry of the problem.

The volume of the liquid in the pool V, is

$$V = \pi (Rh^2 - h^3/3)$$

for $h << R$ the above equation reduces to

$$V \simeq \pi R h^2$$

from this expression the pool depth h can be calculatd as

$$h \simeq \sqrt{V/\pi R}$$

the radius of the pool is

$$r = \sqrt{2Rh - h^2}$$

Subroutining SPILL is called when there is an instantaneous spill of Hexane. The spill initial depth and the radius are calculated from the spill volume using the above equations. Then after each time step the spill depth is reduced by

$$\Delta h = \dot{Z}\,\Delta \upsilon$$

The new spill depth is then used to calculate a new spill radius using the previously defined expression.

For a tank containing 50000 kg of Hexane, the initial pool radius will be 19.5 m. Assuming a constant egression rate of .000083 m/s this pool will burn for approximately 26 minutes. If an uninsulated tank containing propane were located such that it was engulfed by this fire, it would likely rupture.

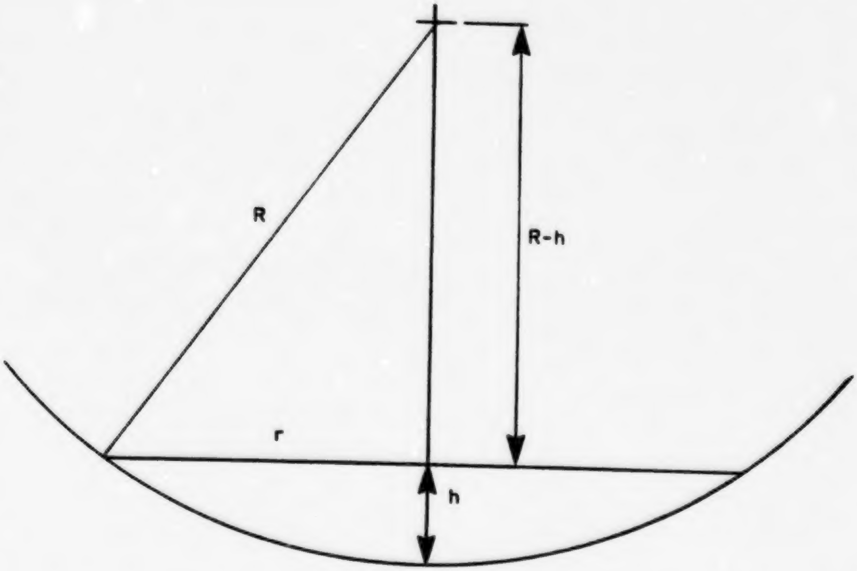


FIGURE D.9-1 ASSUMED SPILLAGE GEOMETRY

The above model is simple and has the basic properties that are needed for the DERACS simulations, i.e.

a) pool fires will have durations that could lead to thermal ruptures if a tank is located in the spill area.

b) the pool shrinks in size with time as fuel is burned away.

c) it satisfies conservation of mass, and therefore obeys basic principles.

The above model can easily be adapted to account for ground permiability. However, the most realistic spill models will require accurate definition of ground elevation, roughness and permiability.

References

(1) Crocker, W.P., Napier, D.H., Quantification of Fire Hazards of Liquid Spills from Tank Trucks, Proceedings of the 4th Technical Seminar on Chemical Spills, 1987, Environment Canada, EN 40-327/1-1987.

(2) Ho Hel, H.C., Fire Research Abstracts and Review, 1, 41, 1959.

(3) Railroad Tank-Car Safety Research and Test Project: Phase II Report on Full Scale Fire Tests, AAR-RPI, RA-11-6-31, AAR-R-201, US6 AFI 17-75 ENG (BK), 1973.

APPENDIX D.10

SUBROUTINE TORIMP

APPENDIX D.10

SUBROUTINE TORIMP

This subroutine is used to determine if any other tanks are impinged by a burning relief valve flare from a rolled tank.

Consider the case shown in Figure D.10-1 which shows a burning flare from one car impinging another.

The calculation begins by determining the distance from the flare source to the center of the other tank, where

$$R = \sqrt{(x'_2 - x'_1)^2 + (y'_2 - y'_1)^2}$$

if this distance R is larger than the flare length L plus $\frac{1}{2}$ l (which accounts for all orientation of the remote tank) then the flare cannot possibly imping the tank, and the calculation ends.

If fire impingement is possible, then the burning flare and the target tank are accurately positioned as shown in Figure D.10-2. With the angles as defined in Figure D.10-2, the points are located as:

$$x'_{P1} = x'_1$$

$$x'_{P2} = x'_1 + (L + r) \cos \theta$$

$$x'_{P3} = x'_2 + (l/2) \cos \theta$$

$$x'_{P4} = x'_2 + (l/2) \cos \theta$$

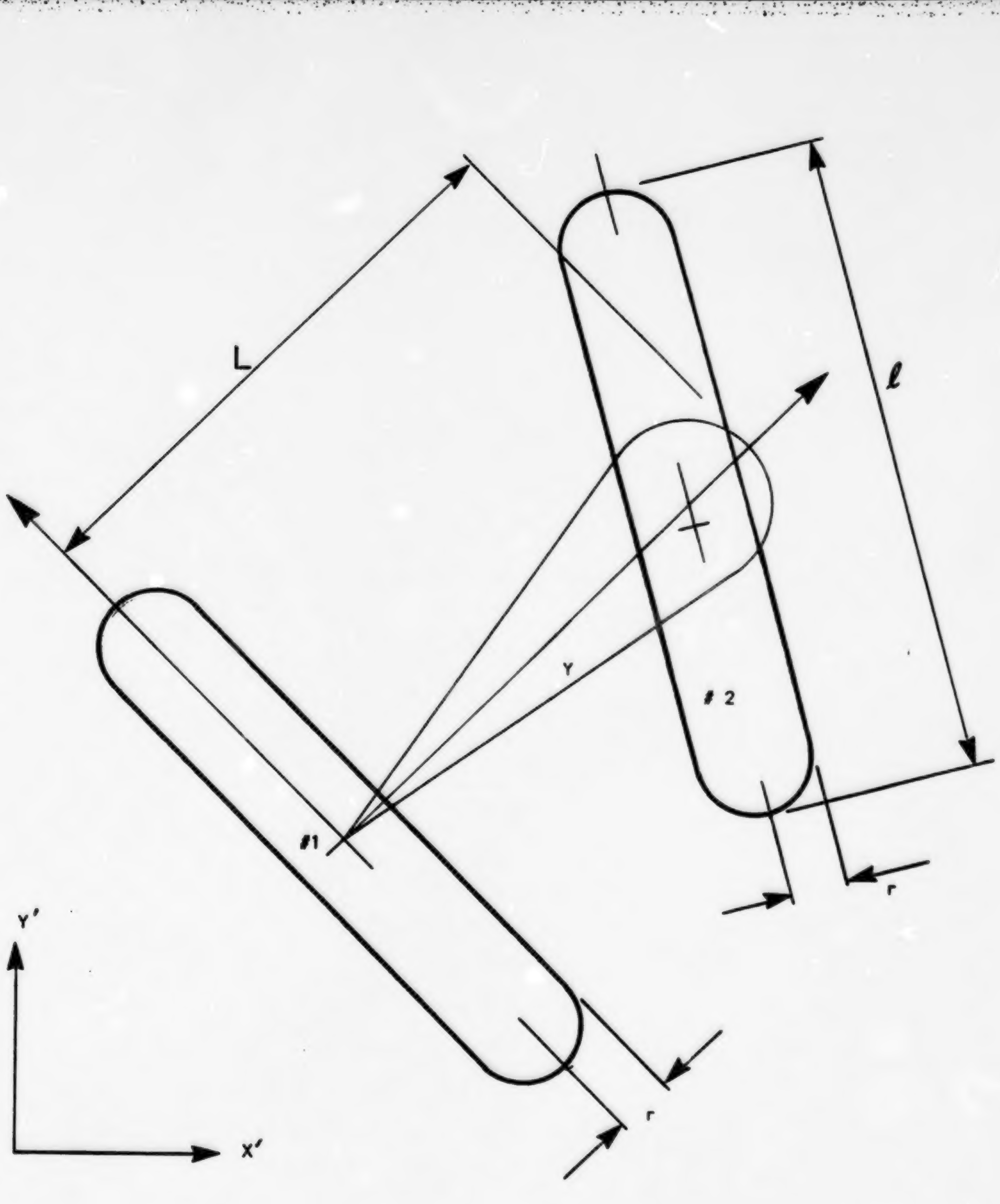


FIGURE D.10-1 SKETCH SHOWING BURNING RELIEF VALVE FLARE FROM ONE TANK IMPINGING ANOTHER

210388 86026F

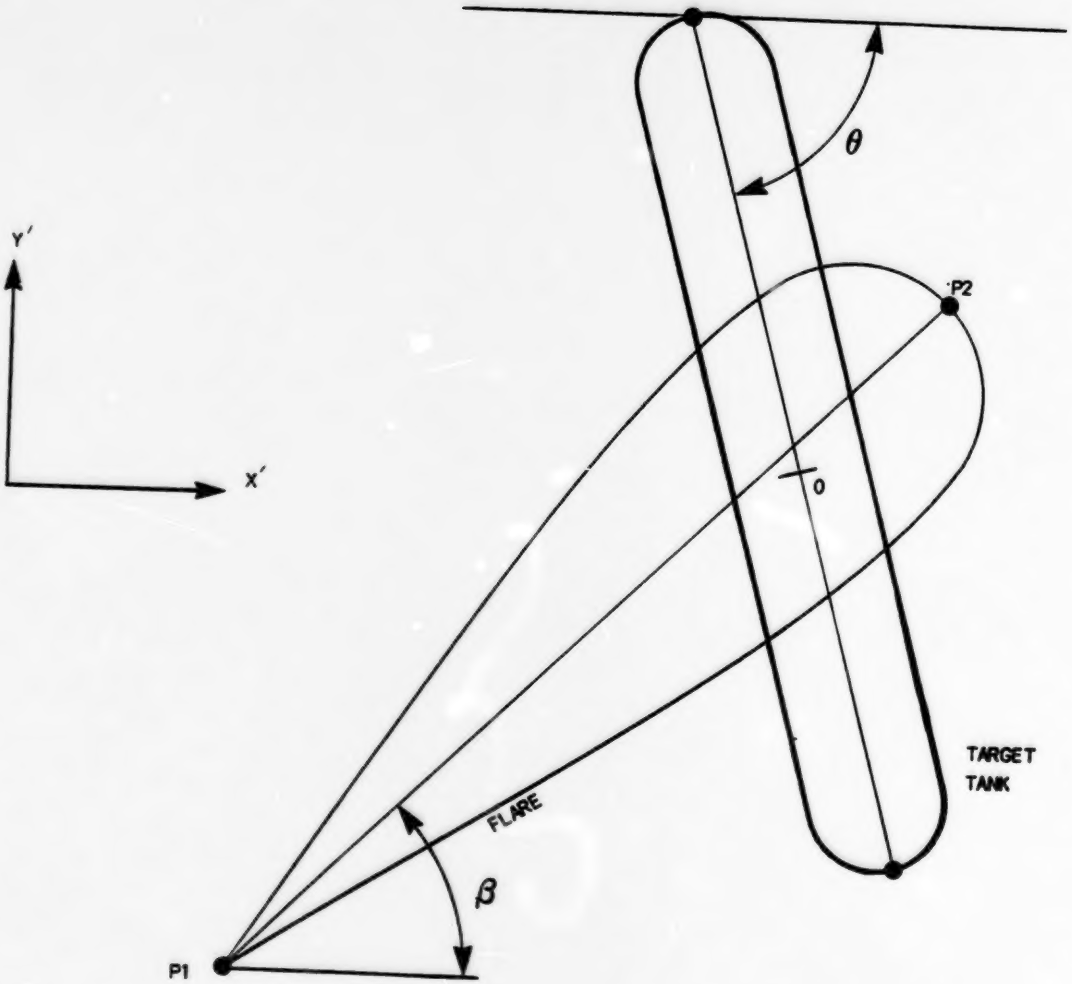

Y′
X′
θ
P2
o
TARGET
TANK
FLARE
β
P1

FIGURE D.10-2 SKETCH SHOWING POINTS USED TO POSITION FLARE AND TARGET TANK

and

$y'_{P1} = y'_1$

$y'_{P2} = y'_1 + (L + r) \sin \beta$

$y'_{P3} = y'_2 + (l/2) \sin \theta$

$y^1_{P4} = y'_2 + (l/2) \sin \theta$

These coordinates are then modified by relocating the origin at the center of the source tank.

$x''_{p1} = x'p_1 - x'p_1 = 0$

$x''_{p2} = x'p_2 - x'p_2$

$x''_{p3} = x'p_3 - x'p_3$

$x''_{p4} = x'p_4 - x'p_4$

and

$y''_{p1} = y'p_1 - y'p_1 = 0$

$y''_{p2} = y'p_2 - y'p_1$

$y''_{p3} = y'p_3 - y'p_1$

$y''_{p4} = y'p_4 - y'p_1$

The coordinate system is then rotated so that the y axis is oriented along the axis to the flare. Figure D.10-3 presents the various angles and lengths used for this rotation

$\theta_2 = \arccos (y''_{p2}/R2) = \beta$

$\theta_3 = \arccos (y''_{p3}/R3)$

$\theta_4 = \arccos (y''_{p4}/R4)$

These angles are then rotated by β so that

$\theta_2 = \theta_2 - \beta = 0$

$\theta_3 = \theta_3 - \beta$

$\theta_4 = \theta_4 - \beta$

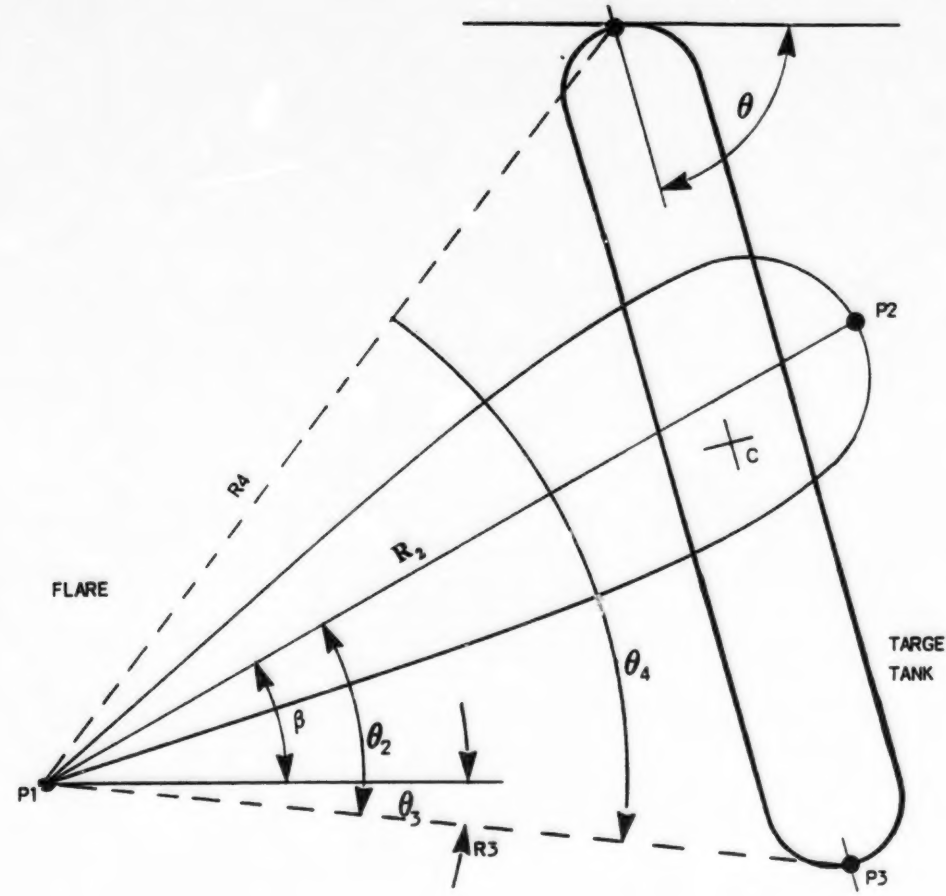

P4
θ
P2
R4
C
R₂
FLARE
TARGET
TANK
θ₄
β
θ₂
P1
θ₃
R3
P3

Figure D.10-3 Definitions of Points and Angles For Rotation and Translation of Coordinate System

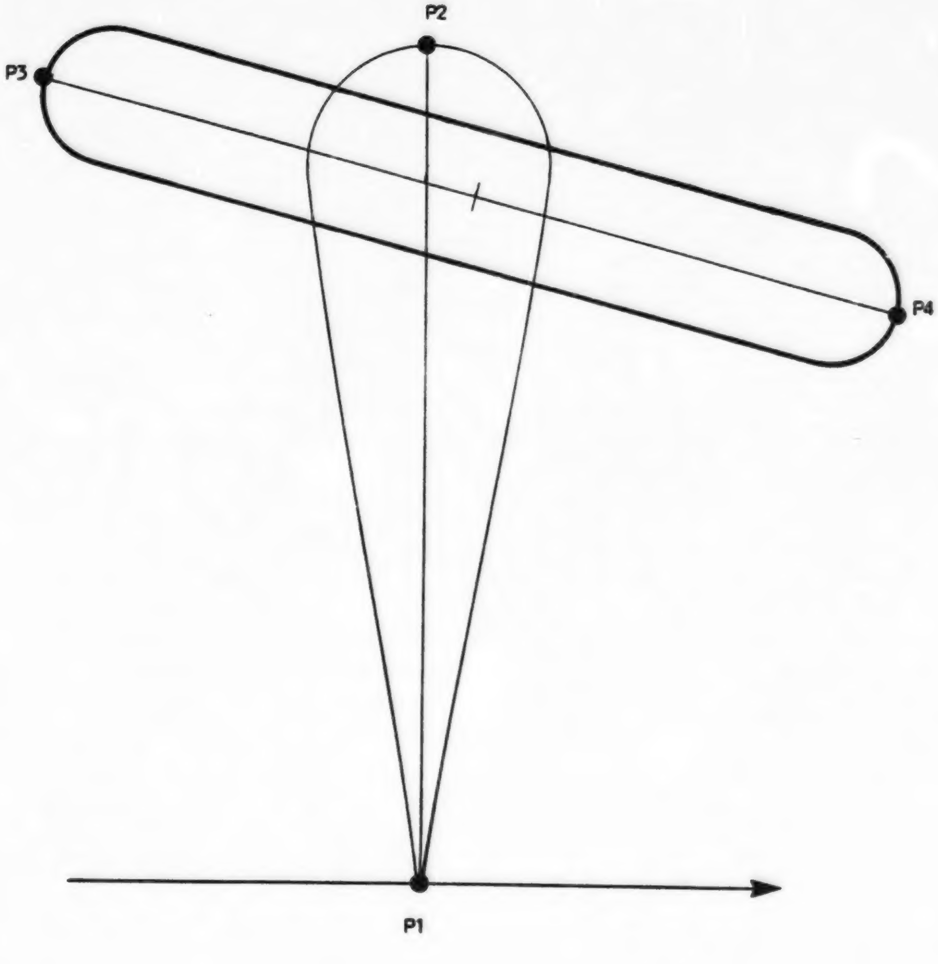


FIGURE D.10-4 FINAL COORDINATE SYSTEM USED FOR DETERMINING FLARE IMPINGEMENT

And, finally, the system points are recalculated for the new coordinate system as shown in Figure D.10-4.

$$x_{p1} = 0$$

$$x_{p2} = R2 \sin \theta_2$$

$$x_{p3} = R2 \sin \theta_3$$

$$x_{p4} = R4 \sin \theta_4$$

and,

$$y_{p1} = 0$$

$$y_{p2} = R2 \cos \theta_2$$

$$y_{p3} = R2 \cos \theta_3$$

$$y_{p4} = R4 \cos \theta_4$$

It is now a simple matter to establish the torch fire impingement based on this final coordinate system.

If P3 and P4 are both on the same side relative to the y axis (and the torch), then fire impingement is not possible and the calculation ends.

If P3 and P4 are both located under the x axis (i.e. they both have negative y values), then fire inpingement is not possible and the calculation ends.

If both YP3 and YP4 are greater than YP2, then again fire inpingement is not possible and the calculation ends.

To get this far in the calculation means that the target tank is straddled across the y axis and if it is close enough to the source, fire impingement is occurring.

The y impingmenet is calculated from:

$$b = \frac{(y_{p4} - y_{p3}) (0 - x_{p3}) + y_{p3}}{(x_{p4} - x_{p3})}$$

If this y intercept is larger than the flare length, then fire impingement is not possible and the calculation ends. If the y intercept is less than or equal to the flare length, then torch impingement is occurring and the impingement distance equals the y intercept.

APPENDIX D.11

SUBROUTINES PLUME AND PUFF

SUBROUTINES PLUME AND PUFF

As stated in the main body of the report, the dispersion of heavy gas puffs and plumes are modelled using a single puff box model. This method was adopted after it was realized that the original concept of using the time dependent SLAB model could no longer be used because of availability problems with the desired version of SLAB.

The puff box model used is that developed by Meroney and co-workers and is fully described in Reference [1]. As with all box models, the puff is modelled as a cylinder which grows radially and vertically depending on air entrainment and heat transfer. The air entrainment takes place through the top and around the sides of the cylinder. Heat transfer takes place by convection from the ground and the surrounding air. The rate of radial growth and cylinder height is based on buoyancy considerations (recalling that the dispersing gas is a dense gas which is heavier than air).

The entire puff cylinder moves downwind and the translational velocity is a function of the ambient wind speed and the shear forces opposing the puff motion.

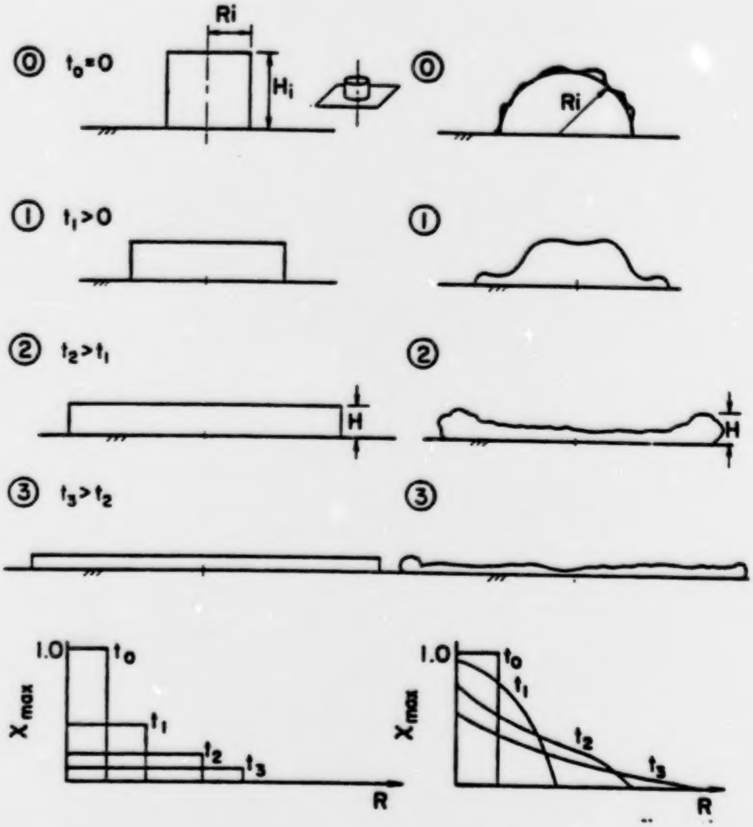
Figure D.11-1 presents a comparison of the appearance of a dispersing puff and the box model representation.

Reference [2] presents the governing equations and also presents validation results based on wind tunnel test data. Box models have proven to be accurate for short times and distances after the release, and therefore are appropriate for the present application.

In this program, the box puff model has also been used to model the dispersion of a continuous release (plume). The plume has been modelled as a series of small instantaneous releases. This was done because of time and budget constraints and should be changed in future versions of the program. Since the puff box model assumes air entrainment from all sides of the cylinder, it will overpredict air entrainment and therefore gas dispersion for the case of a plume. A plume will only have an entrainment through the sides and top which are exposed to the ambient air. This is illustrated in Figure D.11-2.

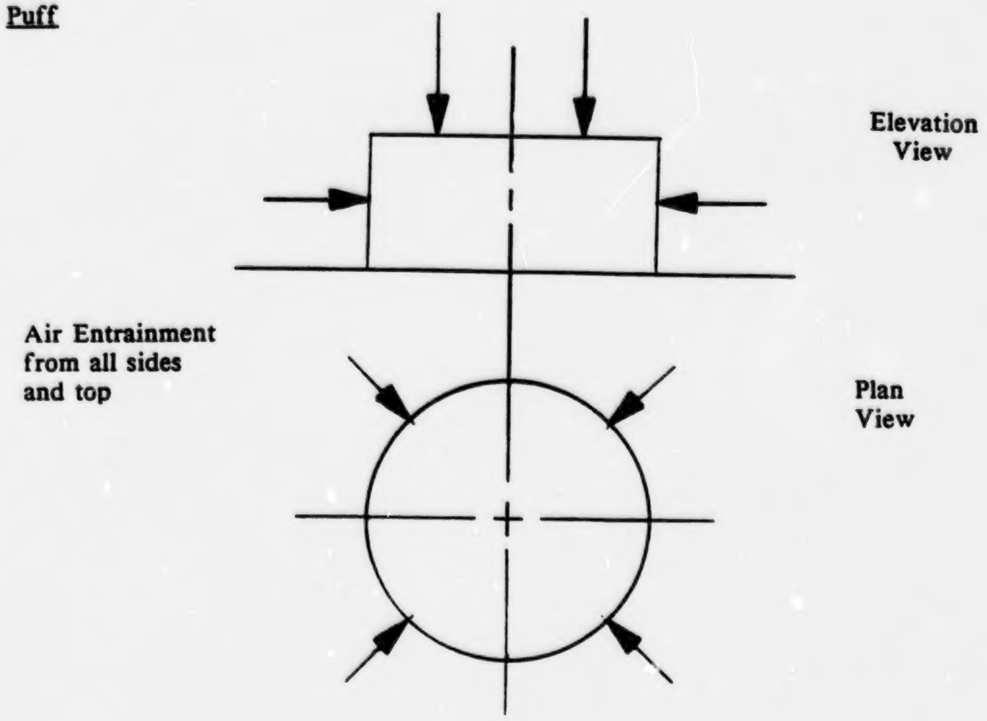
Plume box models exist and could be incorporated in a future version of DERACS. It should be noted, however, that the present representation of the plumes as a series of puffs allows the plume to be modelled if it has a time varying mass flow rate. This is highly desirable for the DERACS simulation. (This is why the time dependent SLAB program was originally chosen).

In DERACS, subroutines PLUME and PUFF are used to calculate the puff radius and downwind distance as long as the concentration in the puff is larger than a preselected constant. These concentrations are as follows for the commodities in DERACS.


⓪ $t_0=0$
Ri
H_i
⓪
Ri
① $t_1>0$
①
② $t_2>t_1$
H
②
H
③ $t_3>t_2$
③
1.0
t_0
X_{max}
t_1
t_2
t_3
R
1.0
t_0
t_1
X_{max}
t_2
t_3
R

WHERE X = CONCENTRATION

Figure D.11-1 Comparison Between Box Model and Real Puff


Puff
Elevation View
Air Entrainment from all sides and top
Plan View

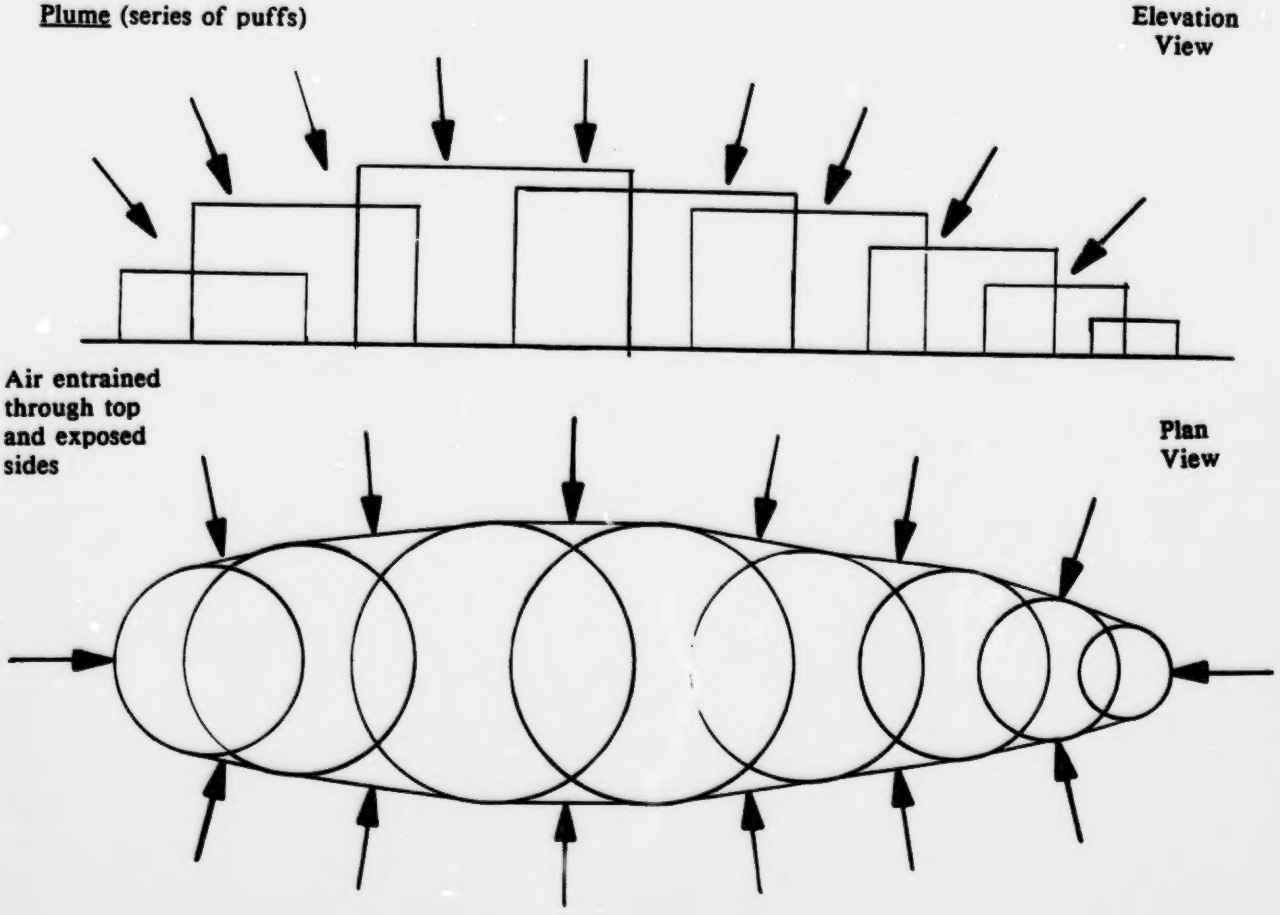

Plume (series of puffs)
Elevation View
Air entrained through top and exposed sides
Plan View

Figure D.11-2 Illustration of Difference Between Plume and Puff Air Entrainment

Commodity	Limit Concentration	Comment
Propane	.038 kg/m³	- lower - flammability limit
Chlorine	.000075 kg/m³	- immediate dangers to life (human)
Hexane	N/A	- does not result in toxic puffs of plumes

The output from subroutines PLUME and PUFF are sent directly to output files named TCARn.PUF where n = 1 - 8 (one for each tank car). Each record in the files contains the time of the puff followed by a number of data pairs (puff radius, downwind distance). The number of data pairs is 300 and each pair are separated by a time step of 20 seconds.

References

(1) Andreiev, G., Neff, D.E., Meroney, R.N., Heat Transfer Effects During Cold Dense Gas Dispersion, Gas Research Institute Report CER83-84-GA-DEN-RNM3.

(2) Meroney, R.N., Lohmeyer, A., Prediction of Propane Cloud Dispersion by Wind Tunnel Data Calibrated Box Model, Journal of Hazardous Materials 8, 205-221, 1984.

Library and Information Centre /
Centre de Documentation (GSLA)
RECEIVED - REÇU LE
JUN 8 1989
Transport Canada / Transports Canada